IMPROVING THROUGHPUT AND PREDICTABILITY OF HIGH-VOLUME BUSINESS PROCESSES THROUGH EMBEDDED MODELING

by

JOSEPH S. DEKEYREL
B.S. Virginia Military Institute, 1986
M.S. University of Southern California, 2003

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2011

Major Professor: Linda C. Malone

ABSTRACT

Being faster is good. Being predictable is better. A faithful model of a system, loaded to reflect the system's current state, can then be used to look into the future and predict performance. Building faithful models of processes with high degrees of uncertainty can be very challenging, especially where this uncertainty exists in terms of processing times, queuing behavior and re-work rates. Within the context of an electronic, multi-tiered workflow management system (WFMS) the author builds such a model to endogenously quote due dates.

A WFMS that manages business objects can be recast as a flexible flow shop in which the stations that a job (representing the business object) passes through are known and the jobs in the stations queues at any point are known. All of the other parameters associated with the flow shop, including job processing times per station, and station queuing behavior are uncertain though there is a significant body of past performance data that might be brought to bear. The objective, in this environment, is to meet the delivery date promised when the job is accepted.

To attack the problem the author develops a novel heuristic algorithm for decomposing the WFMS's event logs exposing non-standard queuing behavior, develops a new simulation component to implement that behavior, and assembles a prototypical system to automate the required historical analysis and allow for on-demand due date quoting through the use of embedded discrete event simulation modeling.

The developed software components are flexible enough to allow for both the analysis of past performance in conjunction with the WFMS's event logs, and on-demand analysis of new jobs entering the system. Using the proportion of jobs completed within the predicted interval as the measure of effectiveness, the author validates the performance of the system over six months of historical data and during live operations with both samples achieving the 90% service level targeted.

ACKNOWLEDGMENTS

I would like to thank my senior leaders within Raytheon, Mike Edwards and Joel Johns for their encouragement, patience and understanding as this process consumed far more of my time and energy than I would have imagined. To Eric Davis, Gene Beauvais, and Dr. Joan Mahoney for their willingness to read, and re-read my intermediate manuscripts and papers – both minimizing the mistakes in grammar, spelling, punctuation, parallelism and helping to clearly say what I intended to convey – any remaining errors are purely mine. To Rick Stikkers, Richard Blum, Carl Davis, Corey Hendricks, Julie Kent and especially to Traci Caldwell for bearing the load of my day-to-day duties when the research and writing got particularly heavy. To my committee members – Dr. Christopher D. Geiger, Dr. Stephanie J. Lackey, and Dr. Mansooreh Mollaghasemi for their willingness to give their time, energy and guidance. Extra thanks to my Committee Chair, Dr. Linda C. Malone for her gentle (and sometimes not so gentle) prodding without which I would have given up several times along the way, and her knowledge and wisdom of the processes, people, and priorities associated with this journey, that lacking, would certainly have driven me mad. To my long-time mentor, Rick Glynn, special thanks for 13 (and counting) years of encouragement and support of my professional career, my education, and the welfare of my family. And to my beautiful wife, Cindy and our wonderful daughters, Ashton and Charlotte for their love and support especially when I was frustrated with the process. And last, though most importantly, I thank God for the modest talents and immeasurable grace he has bestowed on me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACWIP	Actual Current Work In Progress
CORBA	Common Object Request Broker Architecture
CRUD	Create, Review, Update and Delete
DES	Discrete Event Simulation
EDD	Earliest Due Date
FCFS	First Come, First Served
FIFO	First In, First Out
FTE	Full-Time Equivalent
HTML	Hypertext Markup Language
IAT	Interarrival Time
IDD	Internal Due Date
IDIQ	Indefinite Delivery, Indefinite Quantity
JIQ	Jobs In Queue
JSIM	Java SIMulation Library
LCL	Lower Confidence Limit
LIFO	Last In, First Out
MLP	Multi-Layer Perceptrons
MSL	Mean Squared Lateness
NOP	Number of Operations
NWIP	Mean Work In Progress
OAA	Object-Actor-Action
ODD	Operation Due Date
PHP	PHP Hypertext Processor
ProM	Process Mining toolkit
RBAC	Role-Based Access Control
SLK	Constant Slack
SME	Subject Matter Expert
SPT	Shortest Processing Time
SQL	Structured Query Language
ST	Safety Time
SVM	Support Vector Machine
TAT	Turn-Around Time
TWK	Total Work Content
UCL	Upper Confidence Limit
UML	Unified Modeling Language
WEKA	Waikato Environment for Knowledge Analysis
WFMS	Workflow Management System

WIP	Work In Process
XDD	External Due Date
YAWL	Yet Another Workflow Language

CHAPTER ONE: INTRODUCTION

In flush economic times, the elements of excellence that characterize the practice of Industrial Engineering -- reducing cycle-times, decreasing variability, and increasing predictability can mean the difference between a growing business and a struggling one. This is a distinction that can be argued by pundits extolling the competing values of flexibility and control. In leaner times, however, the consequences are more Boolean - the business survives, or it fails. In this latter case, the discussion is rarely friendly and usually not between pundits. It is much more likely to be characterized as a morose recrimination between laid-off employees and their former employers, or between the company leadership and their investors, or in some cases the discussion takes place in front of a judge or an oversight committee.

If the preceding premise can be accepted, why then would companies not uniformly pursue these key performance enhancers provided by rigorous process? The answer is, of course, that these pursuits cost money and are perceived to add to development cost and delivery time without returning sufficient value. In flush times, the argument against process is that this money is better used investing in infrastructure, acquiring key resources or intellectual property (through research or acquisition), or maintaining a strong debt posture. In lean times it is difficult to justify creating processes that add overhead in the face of shrinking margins, falling sales, and imminent layoffs. Part of this prejudice is based on the historical tools and techniques for improving performance often corralled together under the umbrella title of "Systems Management". This set of practices is undeniably successful and is often cited as

critical to the US Space Program. Effective Systems Management implementations added 15% to the cost of a program while returning highly managed risk [1].

When operating in the comfortable embrace of our Nation's generational goal, this trade of cost for minimal risk made good sense. However, after the successful completion of the Space Race the 15% premium was deemed too much to bear for programs concerned simply with developing the next fighter aircraft, the latest main battle tank, or a new nuclear submarine. Through the conclusion of the Cold War, the Systems Engineering approach became ascendant with its slimmer, 10% price point. Now, with the former Soviet Union broken up and the People's Republic of China seemingly more interested in competing in the marketplace than on a battlefield even the 10% cost of Systems Engineering often seems burdensome [2]. Today there are several lower priced alternatives to Systems Management and Systems Engineering, such as Lean Six Sigma which purports to offer sufficient process with a more attractive 5% price point. Today, the twin tines of exploding technology and increasing economic pressure continue to force out even the narrow safety wedge that rigorous process provides. Corporations now routinely operate on the edge of failure, knowing full well that a single slip might precipitate a chain reaction of insolvency litigation that might bankrupt the business and disemploy its workforce. The bewildering proliferation of disruptive technologies force businesses to shorten their response cycles or be lost in last month's technology. The coincident decline of sales means that businesses rarely have the luxury of deep Research and Development budgets to explore leap-frogging technologies as hedges against market

disruption. To reclaim some maneuver room, individuals, corporations and governments have all turned to outsourcing as a way to reduce cost and shift risk. This is a reasonable technique but requires one, significant caveat – the right outsourcing provider must be selected. Selecting an unqualified provider may reduce short-term costs but only because risk has been shifted such that one has limited control over it.

Outsourcing

Outsourcing Source Selection is, in every sense, a critical process – it is vitally important, complicated, and expensive. Whether it is a consumer selecting someone to paint their house, a multi-billion dollar corporation deciding to whom they will outsource their IT support, or the Federal Government choosing a contractor to provide services to the Department of Defense – the choice of source directly affects quality, cost and risk. Source selection is rarely a simple decision made by a single person -- it is usually made within the context of some sort of business process.

A given selection process can be either simple or complex. The simple process costs little in terms of time or effort but often selects less than ideal sources. The complex process is often very expensive but can yield a better source selection. Therefore, balancing the cost of the selection process with the benefit of selecting the most qualified source is vital. The complexity of the most suitable process is very much related to the nature of the product being selected. If one is buying 60-watt light bulbs, then a check of price per hour of life is probably

sufficient, and requires practically no time to execute. Seasoned grocery shoppers perform this type of source selection all of the time. However, when the perceived quality of the product is important, some other (often non-quantitative) criteria must also be applied. As price and complexity increase, the process becomes even more cumbersome both in terms of complexity (more criteria applied), and scale (an outsourcing proposal worth tens of millions [10^6] of dollars can consume the equivalent of thousands of sheets of paper). And there are both fixed and variable costs associated with these types of selection processes that affect how often an outsourcing supplier should be selected, but both corporations and governments seem to have settled on a minimum of a five year term and as long as ten years to make outsourcing viable both in terms of the cost of the competition but also to make the deal worthwhile to the suppliers.

Flexible Ordering Models

This potential reduction in total cost (cost of selection + cost of performance) has led, in recent years, the US Government to move away from traditional, "Full and Open" contract awards (i.e., define requirements, develop specification, seek sources, qualify sources, publish a Request for Proposal, evaluate proposals received, award contract, execute contract, re-evaluate requirements, repeat). Instead, the government has exhibited a preference for IDIQ (Indefinite Delivery, Indefinite Quantity) contracts where the sources selected are capable of performing across a broad scope of possible tasks. The corresponding construct in the

commercial market are sometimes termed “Blanket Ordering Agreements” or “Blanket Purchase Orders”. These IDIQ awards may be to a single contractor or to multiple contractors. They are typically executed over a span of many years and may have very high funding ceilings (some measured in billions [10^9] of dollars). Within these contracts (because of the pre-negotiated rates) the task award process for a specific piece of work is considerably more streamlined than the traditional full and open competition. This can result in a more efficient use of the government’s funds with more of the funding available to do work because less is expended in the competition process. An additional benefit to the agency requiring goods or services, often cited in their justification for the acquisition, is that the agency can expect that their delivery times would be much shorter.

Though the program that will provide the context and data for this research also provides support to the US Government, the techniques and attendant benefits described are applicable to any organization that provides, or seeks to provide, broad-scope, outsourcing services.

Context System

A recent single-award IDIQ contract was awarded to a major defense contractor to support world-wide training operations. The award has a ceiling on the order of $10 billion over a 10 year period. A base task order to that award accounts for approximately 20 percent

of the $10 billion ceiling. The remaining 80 percent of the ceiling is set aside to cover additional task orders within the broad scope of work awarded.

As the senior managing engineer for the recipient of the above award, the author has been deeply involved in the development of the extended business processes required to support this new contract vehicle. Central to the IDIQ contract type is a business process to handle order management. This order management process runs across the entire enterprise from field customer, through contracting agency, and then through nearly all of the various functional and operational sub organizations of the contractor team. The quantity of dollars was mentioned above, but perhaps the more telling metric is the number of individual orders that must be processed by the combined government-contractor team. During the first two years of execution, the business processed on the order of 1,200 task orders per year (100 per month, or 5 per business day). If the orders were for books or other packaged consumer goods, then an Amazon-like model could have been used. If the orders had been for cars – allowing for some limited buyer configuration, then a model based on the auto industry might have been appropriate. And while there is some limited similarity to the construction industry, the price variance for the individual orders is greater than five orders of magnitude. The author has been tasked with developing the system to responsively schedule the *delivery* of the proposal, *not* the product itself.

From a historical perspective, the resources required to execute the corresponding process on the preceding contract amounted to “X” Full Time Equivalents (FTEs). As the volume

of orders has increased by nearly 1,500% one might expect the upper bound on the resources required would be on the order of 15X FTEs. By implementing some of the techniques described in the following, the upper limit on resources has remained under 2X FTEs. Even at service industry rates, for an effort of this magnitude this reduction in required resources equates to millions [10^6] of dollars annually. This process and system for capturing the remaining ceiling onto this IDIQ contract will provide the practical backdrop for this research.

Generalized Problem

Recast in more general terms and in a broader sense, this process begins with the identification of a requirement by a *consumer* and contracted by his *agent* through a *broker*. It ends with the work completed on-time, the final bill submitted, and the *broker* as well as its *suppliers* paid an amount less than or equal to that proposed. Each of these orders requires considerable effort by both the *sales agent* and the *broker* with statements of work and order packages being developed and staffed by the *sales organization* and proposals being generated by the *broker* and his *suppliers*. The proposals must be reviewed and, if accepted, activated by the *agent*, and then subcontracted, in many instances, to existing or new *suppliers*. This flow is depicted in the sequence diagram at Figure 1.

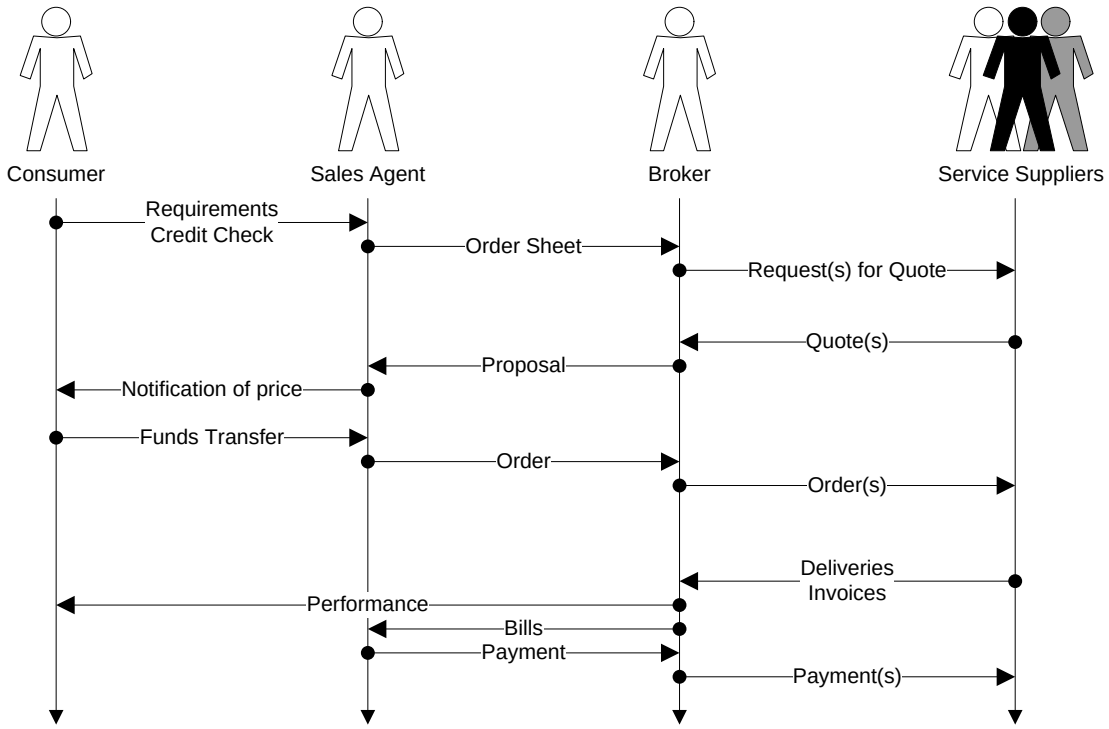


Figure 1 - UML Event Trace Diagram

As a complement to the existing project management literature which covers the performance segment of the diagram above, the existing supply chain literature that covers the Order and Deliveries sections, and the existing finance and accounting literature that covers the Invoicing, Billing and Payment sections, the area of focus for this research will be on the order and proposal sections of this process, as shown in Figure 2.

While this diagram is representative of the aforementioned government contracting vehicle, it is noteworthy that a custom home builder might draw a similar diagram with the consumer being interested in a custom home, the sales agent filling out the order form

describing the customer's desires, and the broker (general contractor) evaluating the order (with its suppliers) to determine a final cost and schedule for the home.

Based on historical data for the test system, these orders *must* flow (in a steady state, steady flow process sense) through the consolidated order and proposal process at a nominal rate of five per day. As a starting point, a notional (without analysis) cycle time of 21 calendar days was set. It is this cycle time that most neatly captures the efficiency of the combined Agent-Broker-Supplier process.

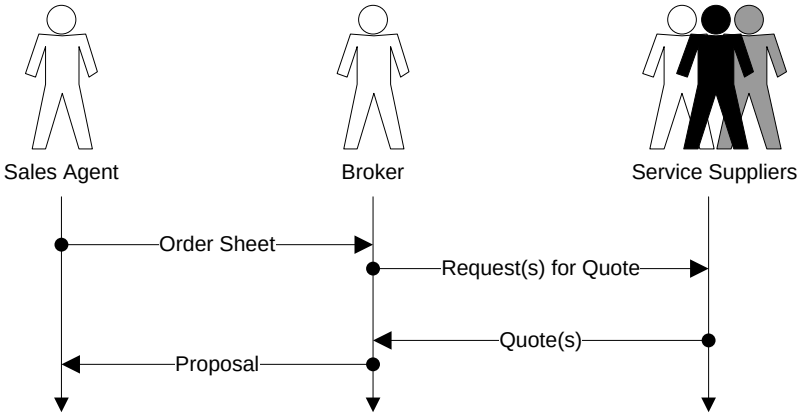


Figure 2 - UML Event sub-trace for subject system

Unfortunately, measures of central tendency tell very little of the story in this situation. Lower than required mean cycle times on the order of 10 days obscure the variance, with minimum cycle times and maximum cycle times differing by at least two orders of magnitude. Traditional, manufacturing-centric, quality-based wisdom suggests that this variation should be vigorously stamped out by eliminating the *sources* of variation. And, in fact, many of the reducible sources have been and continue to be attacked. However, the single largest

contributing factor to variability is the scope of the order being processed. Orders are taken whose final proposed values vary between hundreds of dollars and tens of millions of dollars. Further complicating the management of these order efforts are a significant collection of variables including requirement completeness, requirement maturity, mission lead time, task complexity, risk (cost, schedule, and performance), task order type (Time and Material or Fixed Price), location where work is to be performed, customer priority and many others. Some of these are neatly captured on the order sheet provided by the Sales Agent, but several, most notably those respecting requirements, only become visible upon semantic review of the supporting documents.

Now, finally recast as a generalized problem – there is a flexible flow shop in which the stations that a job passes through are known and the jobs in the stations queues at any point are known. All of the other parameters associated with the flow shop, including job processing times per station, job value, and station queuing behavior are uncertain though there is a significant body of past performance data that might be brought to bear. The objective, in this environment, is to meet the delivery date promised when the job is accepted.

With the continued success of the contractor hanging in the balance, the author proposes to perform data collection and analysis to measure the performance of the existing business process, build a suitable model of the process as a baseline of comparison, and then develop an embedded process model coupled with nondeterministic algorithms to improve predictability of this critical process.

Prior Work

Prior to the start of execution of the contract, which employed the author in 2008, the procuring government agency approached the author, and asked that the author develop a tool to assist the agency in tracking its internal business process for generating and issuing order sheets. The author agreed and developed the initial version of that tool. While developing this tool for the agency to use (though developed at contractor expense and hosted in the contractor's data center) the author recognized the logical benefit of extending the tool to encompass the contractor's emergent business process for preparing and delivering proposals in response to the agency's order sheets. The generalized nature of the process for developing such a system is depicted in Figure 3.

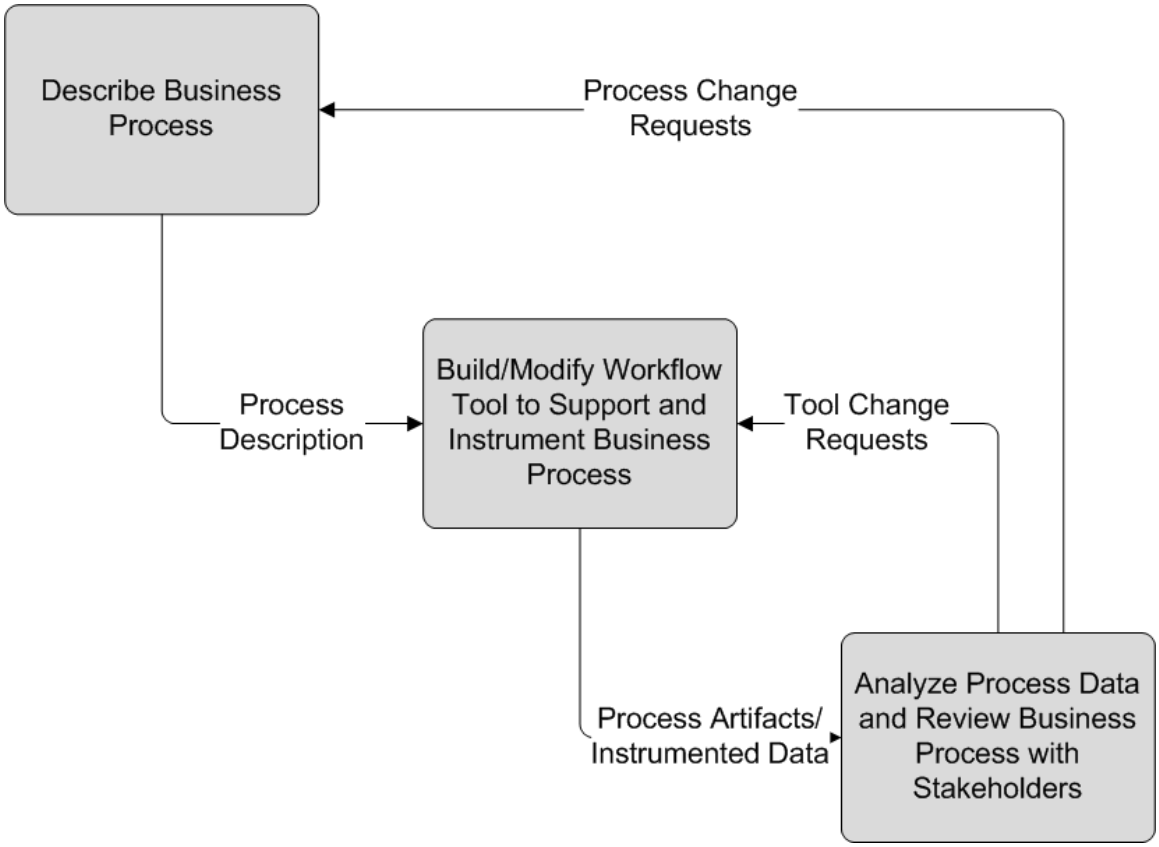


Figure 3 - Initial phase of business tracking system development process

The author recognized the potential power of gathering transactional data as the documents were executed through the paired agency-contractor processes. It was with these goals in mind that the author wrote the initial workflow tool and documented the corresponding initial business processes. Central to this workflow management approach was the accountability that is enforced by the transactions that are recorded - for each change in state of the affected document, the date and time of the change, and the person effecting the change are recorded. [3]

After the process and tool had been running for six months, a consolidated performance review was held. During that meeting questions were raised by the agency about order processing times. The author was able to answer in objective terms about the mean processing times by individual process step and by the category of services provided. The results, while not necessarily pleasant for all involved were both illuminating and beneficial. To allow greater visibility, the author then had this processing time report recast as a multi-segmented bar chart (Figure 4) which was made continuously available to all parties within the agency and to the contractor.

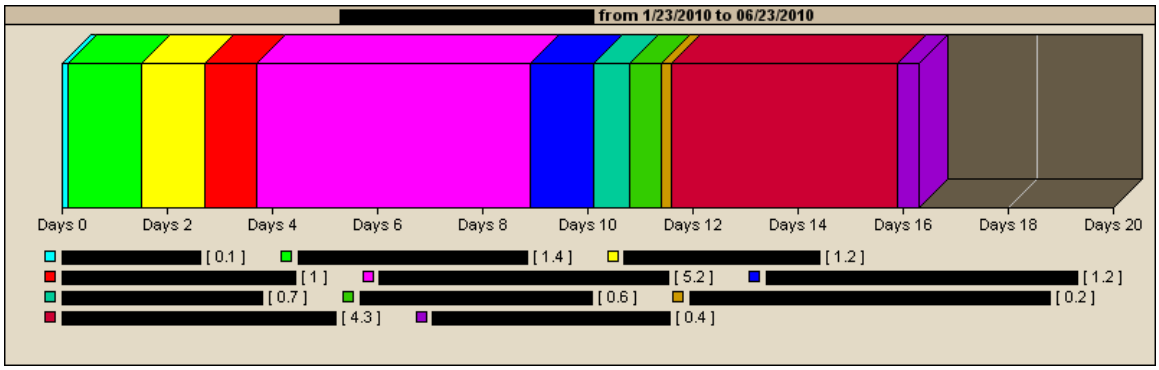


Figure 4 - Multi-segmented bar chart of processing time

The positive effects of this transparency were remarkable to both the contractor and the customer. Simply measuring what was being done by both the agency and the contractor, and making the results continuously visible through a web-based tool drove combined processing times (inclusive of agency processing and contractor processing) from 88 days down to 48 days without applying any particular pressure to any point of the system. Overall

efficiency was improved as well as customer satisfaction by making the answer to most questions regarding status immediately available from any web enabled computer.

During the subsequent two years, the process and the tool have continued to receive updates to streamline processing and allow for additional categorical information to be captured on the order sheets. It is on this now firm foundation that the author proposes to build an engine that will allow for accurate prediction of proposal delivery time for new orders. The overall process for this research is presented in Figure 5.

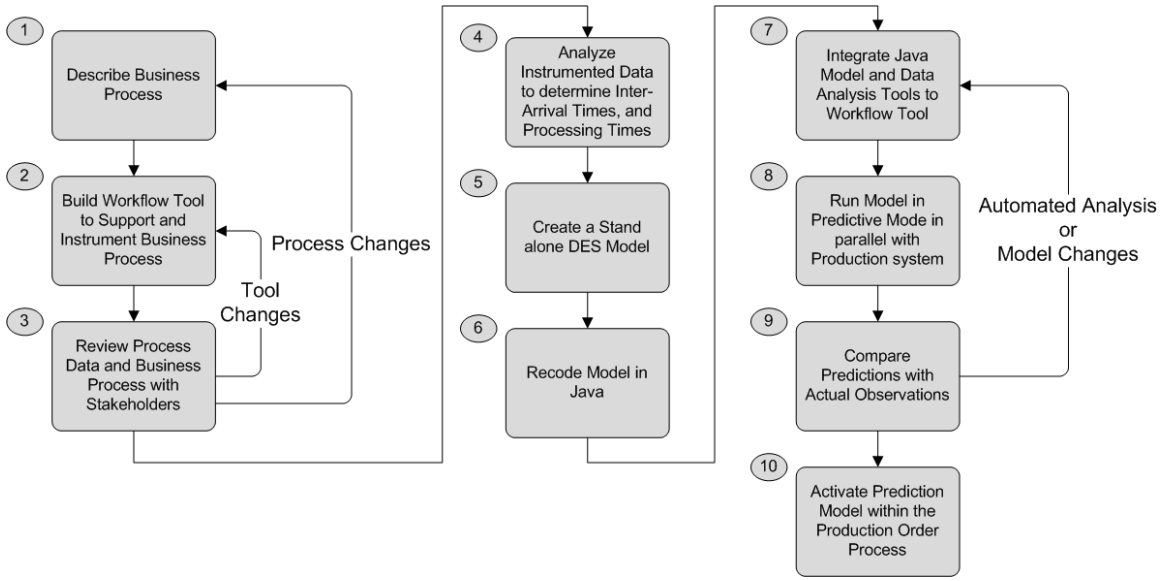


Figure 5 - 10-Step system development process

Research Objectives

The overall goal of this research is (1) to extend the literature with respect to the use of embedded modeling and automated data mining to enhance predictability in uncertain

processes while specifically addressing techniques for dealing with realistic tracking and performance data through output-ordered queue analysis and empirical queues, (2) develop a prototype embedded model combined with automated analysis techniques to improve the predictability of a representative, multi-tier business process with dynamic behavior, and (3) conduct a feasibility study of these techniques by deploying the prototype into a production environment to validate the benefits of the combined process on predictability. At the conclusion of the research, a proficient practitioner should be able to apply this approach to similar multi-tiered, electronic workflow management systems. For the researcher, the non-standard queuing components, both for analysis and simulation, should provide fertile ground for the exploration of system optimizations outside the well studied First Come-First Served (FCFS) queuing policy.

Subsequent Chapters

As the development of the theoretical and practical portions of this research have been intertwined with the author's professional pursuits and portions of the results have already been published, accepted for publication, or submitted for publication, an alternative organization of the remaining chapters is utilized. Chapter Two describes in detail the proposed 10-step process presented in Figure 5. Chapter Three describes the proposed heuristic algorithm developed to decompose the event logs from the test workflow management system (WFMS). Chapter Four contains a published paper in which the author begins the logical

argument for a new approach to Due Date Quoting in complex systems, particularly those that do not use an FCFS queuing policy. This chapter focuses on Steps 4 and 5 of the 10-step process. Chapter Five contains a paper, currently in review for presentation, that presents the development of the prototype implementation of this new approach (Steps 6 and 7) and the initial predictive results. Chapter Six contains a paper, accepted for publication, which extends the results from Chapter Five by incorporating an error distribution into the predictive process against historical workflow orders. Chapter Seven is a paper, submitted for publication review, which details the results of the predictive prototype in setting accurate due dates for a production order processing system – completing Steps 8, 9 and 10. Conclusions are drawn and suggested areas for further research enumerated in Chapter Eight. The gap in the existing literature is bounded in the extended literature review in Appendix A. A listing of practical considerations discovered while following the proposed 10-step methodology is provided at Appendix B.

The reader should note that Chapters Four, Five, Six, and Seven are written as standalone papers and included in their entirety. As a consequence, certain material in these chapters (i.e., mathematical formulations, descriptions of the test system, etc.) are redundant.

CHAPTER TWO: RESEARCH METHODOLOGY

The overall goals of this research are (1) to extend the literature with respect to the use of embedded modeling and automated data mining to enhance predictability in uncertain processes while specifically addressing techniques for dealing with realistic tracking and performance data through output-ordered queue analysis and empirical queues, (2) develop a prototype embedded model combined with automated analysis techniques to improve the predictability of a representative, multi-tier business process with dynamic behavior, and (3) conduct a feasibility study of these techniques by deploying the prototype into a production environment to validate the benefits of the combined process on predictability. To achieve these goals the author executes the tasks summarized in the 10-step flow diagram in Figure 6, provides the graphical outline for the remainder of this chapter.

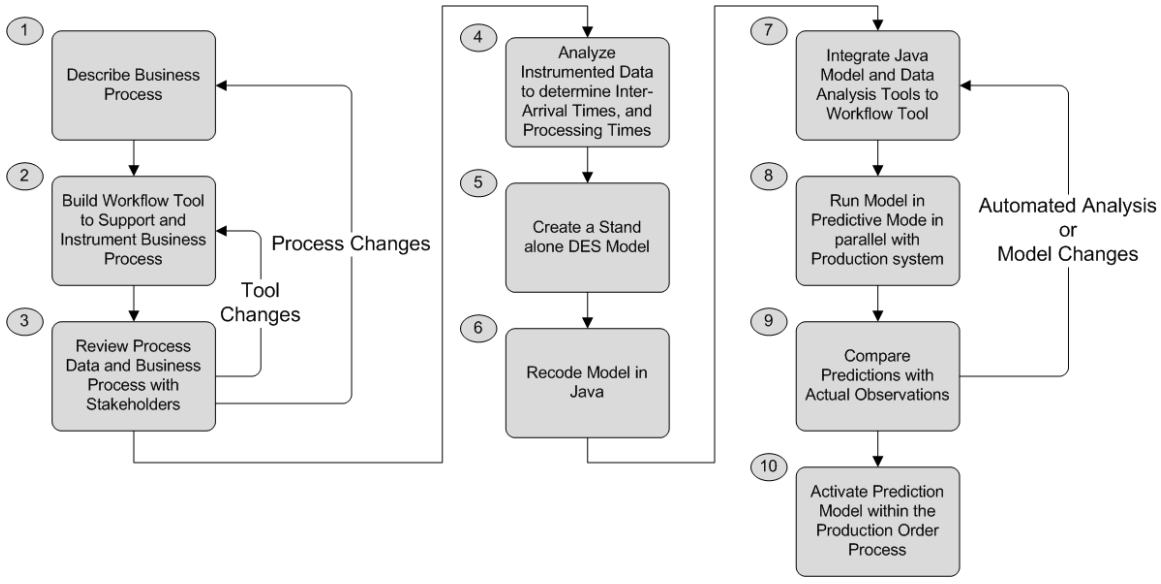


Figure 6 - Embedded model development process

At the end of this 10-step process, the author expects to have a system capable of answering the specific question: “Given the descriptive attributes of an object and the current state of the business process system, when will the object exit the system with 90% confidence?” Though the question is stated in terms of a 90% service level, the resulting prototype could as easily be tuned for other services levels. Throughout this chapter, a series of graphics highlight the portions of the system under study or development and their interrelationships. This series culminates with a pictorial representation of a system the author offers that is capable of answering the question above.

Step 1 - Describe the Business Process

In order to begin the process described in Figure 6, the practitioner must first capture the business process in question. Depending on the circumstance, the practitioner will have some degree of familiarity with the process to be captured. This familiarity will range from highly familiar in the case of an in-house practitioner to limited familiarity for a consulting practitioner. In the case of the in-house practitioner, one should be careful not to assume that tenure equates to understanding – 15 years of requisitioning light bulbs does not qualify the practitioner as an expert in the Supply Chain functions of ordering, receiving, and stocking light bulbs. Similarly problematic, the consulting practitioner should avoid forcing the process now under consideration into a previous consultancy’s pattern as a shortcut to developing the process documentation required.

There are several different representations that can successfully be used to capture business processes and much literature exists citing the virtues of one scheme over another. Any representation that can capture the behavior of the process in question is sufficient. This author is primarily concerned with capturing the process as efficiently and unobtrusively as possible. To that end, a sufficient modeling paradigm that the practitioner is well experienced with is as important as any other concern in selecting a representation. As this author is most experienced with an Object-Actor-Action (OAA) framework, it is that framework that will be used for capturing processes and their behavior throughout the research.

An OAA diagram is analogous to a Unified Modeling Language (UML) Activity Diagram combined with a UML Use Case diagram to indicate the actions taken on an object but also annotating the user (or user class) responsible for the action. It also captures status changes to the object as the process progresses. The opening phrase of the subject process is provided as a brief example in Figure 7. In this example, the object that flows through the workflow system is an “Order Sheet”.

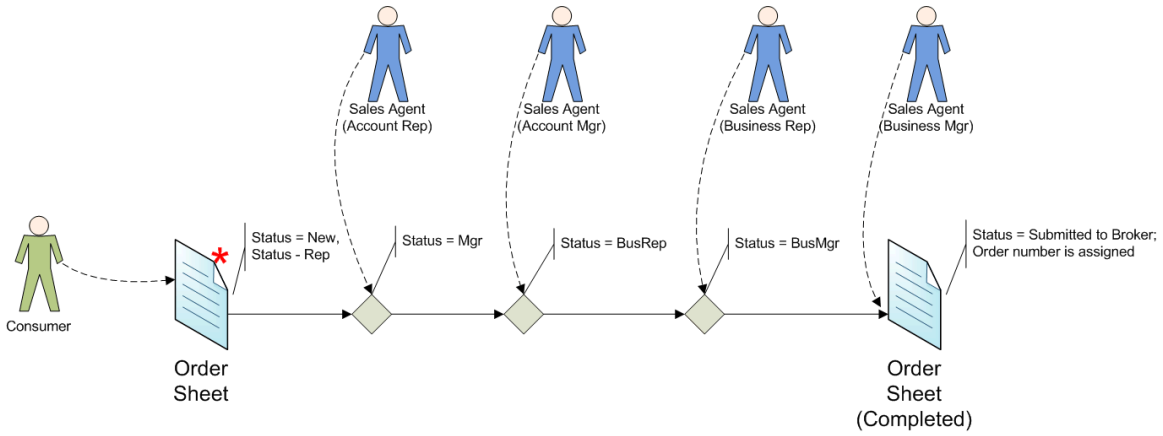


Figure 7 - Sample Object-Actor-Action diagram

The author has found that an appreciation for the overall process is useful before delving into the details of the individual steps in the process. This activity of capturing the top-level business process is often chaotic as complex organizations may not have an internally consistent view of how they conduct their own business. This tends to lead to confusion and often contention among the functional and operational Subject Matter Experts (SMEs) gathered to assemble this artifact. When complete, this top-level view of the process quickly highlights the interpersonal and inter-organizational interfaces. Perhaps of even greater value is the highlighting of significant gaps (lack of interface) between organizational elements. These gaps usually become obvious as places where the diagram is discontinuous.

As best stated, "The greatest leverage in architecting is at the interfaces" [4]. The things that traverse these interfaces are the objects that the process functions upon - usually documents (whether paper or electronic) in the business process context. The enumeration of the interfaces will usually provide the statuses applied to the objects (usually written in the past

tense, e.g. “Supplier Quotation(s) Received”). Similarly, the people processing the objects between the interface steps are the actors.

It is these interfaces that must be clearly understood when capturing the attributes of the object(s) that are processed. With a draft top-level process view established, the individual actors in the process can then be efficiently interviewed to confirm the object attributes that they require in performing their step(s) in the process. It is important not to lose sight of the individual actor’s actual requirements when attempting to consolidate their collective inputs as this is a sure way to end up with not only disenfranchised users but potentially uncontrolled portions of the business process as well [5].

Along with the attributes of the objects, the responsibility for managing the objects must also be captured. For any non-trivial business process executed by a complex organization there will be some separation of duties and responsibilities which will be referred as a role in this context and denotes authority or permission across all instances of various object types. Disentangling these ownership/authority boundaries is a challenging practice, the consequences of which are noted by Larsen and Klischewski [6]. As the resources (number of actors) applied to a process increase it is sometimes advantageous to associate particular users with particular objects (a sales representative with a particular customer, etc.). This is described as an assignment and would denote authority or permission across the subset of object instances to which the actor is assigned. With these definitions, the practitioner can then determine the object permissions by role and/or assignment. The author prefers to use the

CRUD method [7] to catalog which users or user classes have permission to Create, Read, Update, and Delete instances of objects in the system though other methodologies could certainly be employed.

While all of the above activities may seem to be focused in the virtual world, it is critical that the corresponding physical portions of the process are not overlooked. During the interviews the practitioner must elicit all of the actions (real or virtual) required after each step in the process – from sending a notification email to physically stamping a paper form, each of the actions must be cataloged so that the potential of leveraging automation systems (Enterprise Resource Planning, etc.) can be explored.

The final step in this cycle of capturing the business process is to consolidate the data collected and review it with the collected set of actors.

Step 2 - Build a Workflow Tool for the Business Process

After the process under study is well understood and well described the next step is to build an information system to facilitate the measurement of that process. To construct such a system three phases organize the task manageably: database design, application development, and report creation. In this context "Database Design" encompasses both the logical design of the database (its abstract schema) as well as the instantiation of that abstract schema as a concrete schema against a particular database engine. Similarly "Application Development" encompasses evolution of the architectural elements (logical, security, network, and physical),

which informs the technology selection, which underpins the actual encoding of logic in the specified language. And while listed last, the "Report Creation" phase must remain fixed as a design driver throughout the database and application development processes – ensuring the database design and the application developed readily support the required reports.

Define the schema to contain objects – during this portion of the logical database design process the practitioner creates a table for each distinct object type discovered within the business process elicitation. In addition to a unique, system generated, primary key for each object, all of the attributes associated with the object are encoded either within the table itself or within companion tables (when the objects and attributes form a "one to many" relationship). Care must be taken to thoughtfully establish which attributes are required and which are optional. This partitioning may change as the object changes state. For example, a new order request object may simply need a customer and a sales representative (in addition to its unique key value, and its creation date – both system generated) to allow object creation, but will certainly require that additional attributes are populated before it can be submitted for bid processing.

Define schema to capture transactions – this extension of the database design process extends the object schema. The object schema alone simply reflects the state of the system at the current time but as a "workflow" system, it must account for time as well. Transactional tables are a means of reflecting time (in this usage – history) within a database. At their simplest, transactional tables provide a framework to record the identification (ID) of an object,

the date of the transaction, which actor effected the change in state, and the new (or old) status of the object. Depending on the network and physical implementation of the database and application it may be desirable to disambiguate the order of transactions with a system generated unique key on the transactions as well.

Define schema to identify actors – this portion of the database design defines the structure that will represent all of the entities that can change the status of an object or who are interested in such changes. A user key, user name, and email address are required in such a schema though many other attributes are likely to be desired such as a phone number (or numbers), physical addresses, company affiliations, and so on.

Define schema to identify actions – if the practitioner has been thorough in understanding the business process, this schema should be simple to define and the data to populate simple to create. In the author's implementation, the table that implements the action schema has three fields: a unique, numeric ID, and abbreviation for the state of the object, and a long description of the object state. These states, or statuses, are simply the enumeration of the steps in the business process as elicited in step 1. Depending on the number of unique object types to be handled in a system it may be possible to use a single table to reflect the statuses of multiple object types, though this small efficiency is likely outweighed by the added complexity of keeping the statuses of the various objects out of conflict.

Define schema to support security – this step in designing the database is simple to overlook, especially for a small implementation, but failure to adequately address this aspect of the overall system within the database schema will cause the application portion of the system to be much more cumbersome to develop and maintain than necessary. As described in step 1 above, security can be based on roles, assignments, or both. The simplicity of implementation for purely Role-Based Access Control (RBAC) scheme can lead to unauthorized access when users are promoted to overcome RBAC limitations [8]. There are several sufficient patterns that can be used to provide a security structure. The criteria involved in choosing an appropriate pattern are associated with the number of actors using the system, the availability of an existing user directory external to the application, the geographic and organizational diversity of the collection of actors, and others. Successful patterns may, based on the criteria above, range from a local user table with application enforced credential policies to enterprise-wide directories containing both internal and external actors. As the security aspects of a well designed system tend to pervade the implementation, care should be taken to account for the potential growth of the process – a process that today might run comfortably with a dozen actors in one warehouse might be vastly inadequate when the process scales to run with hundreds of employees located at several geographically dispersed locations.

Normalize schema – as a routine part of any database design, after an initial, logical database schema has been developed it should be normalized to minimize data redundancy (a seasoned practitioner may perform this task sufficiently during the development of the

individual portions of the schema above such that this step is simply a validation of normalization). In addition to the database-centric benefits of normalization, the author finds the hands-on process of normalizing the schema to be of value in pre-defining data to be entered – thus minimizing the quantity of free text entry required (or allowed) in the application.

With the database schema logically defined and implemented against a database engine, the next task is to define the application environment that will implement both the business logic of the business process and controlled access, in accordance with the CRUD matrix, to the object store. As the strategic and tactical requirements, corporate security strictures, and customer infrastructure details that informed the larger architectural development are beyond the scope of this research, a short summary of the salient points is appropriate – external users should have the same experience as internal, no software beyond a web-browser could be assumed on the client machines, and thousands of users should be expected across (nearly) every time zone. These points lead quickly to a web based solution with some form of server-side scripting.

Of the many choices available (Java Server Pages [JSP], Active Server Pages [ASP], and PHP: Hypertext Preprocessor [PHP] to name a few), PHP was selected as the server side language for this project based on the author's familiarity with it. It was coupled with Microsoft's SQL Server as the database engine – similarly, any database engine capable of handling the required transactional loading could be used. Both internal and external users

make use of Microsoft's Internet Explorer as their web browser which serendipitously shortened the development effort by obviating multi-browser integration issues. These tools were required to interact with the Enterprise user directory necessitating the creation and management of internal and external user accounts in a single data store.

With the above architectural decisions made, the next task was the creation of the web application itself. A cursory review of the requirements of such an application demands a consolidated list of objects, a view to add an object, one to edit an existing object and a read-only view. The practitioner is then faced with the choice of hand coding the pages or using a third party tool to generate the pages described above. Based on perceived framework flexibility the author selected a code generation tool that produced PHP pages based on the already defined database schema. The choice to use a code generator saved many hours of HTML and PHP development, however the constraints of placing the workflow code within the tool's required framework may have outweighed the time savings associated with the automatic page generation. The tool selected was eventually extended by its author to support role-based security based on a local user table. This capability was found to be insufficient and was subsequently replaced by a hybrid integration combining locally defined roles and assignments in conjunction with enterprise user and credential management. All of the available code generation tools reviewed by the author worked directly on the object tables, none of them supported any sort of automation to facilitate transaction or audit history creation – these capabilities must be added by the practitioner.

As the application is being developed it is appropriate to begin to formulate the queries and display layouts for an initial set of reports. Some reports are obvious such as count of objects by current status, objects by actor assignment, and objects by creator. Other reports are more subtle and might provide insight into things like the time spent by objects in various statuses.

Thoughtfully considering the perspective of the various actors may lead to additional aggregating attributes. As an example, each of the actors in Figure 7 is a distinct customer from the broker's perspective, and each would likely desire a differing aggregation. The Sales Agent would like to see orders for all of his clients, while the Business Manager might want to see orders by product, or Sales Agent, or payment terms. There are cases where the attributes will be hierarchical – each of the Account Representatives works for only one Account Manager. In other instances, the attributes will not align with organizational boundaries as in the case where clients may work with multiple Account Representatives for different products. The key is to remain flexible to differing reporting (especially aggregation) requirements depending on the customer's perspective.

The WFMS that serves as the test bed for this research implements all of the architectural and design consideration described in this section. Critical to the analysis step (Step 4), the test WFMS implements transaction logs that are written out to the WFMS repository (a Microsoft SQL Server, in this case). These transaction logs record the arrival times and locations for each order as it transits the system.

Step 3 - Review the Process Data and the Business Process

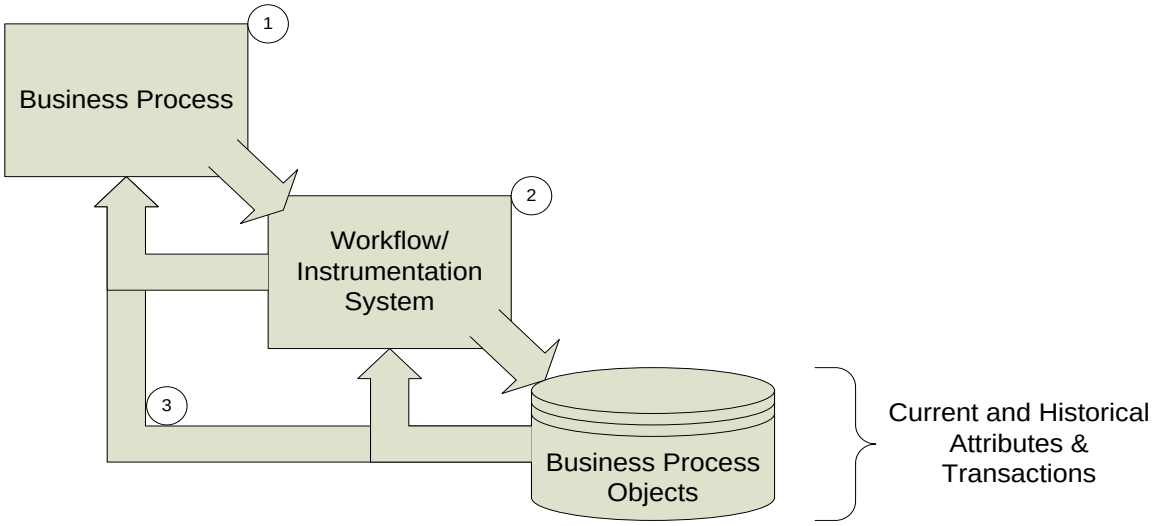


Figure 8 - Phase 1 of the system development

With the business process in question well defined, documented and communicated and the supporting tool developed and tested it is appropriate to run the workflow tool in a production environment. The reader should note the feedback arrows from Step 3 back to Steps 1 and 2. Irrespective of the time and effort invested in performing Steps 1 and 2, there will be issues that arise when the process/tool combination is put into production.

The key to this third step is to look at the data *frequently*, and talk to the users *frequently*. The point of doing so is to make sure the system, the data and reality match, if they do not the practitioner must modify the process, the tool, or both until they do.

As a result of this step, the author implemented several changes but as examples, consider the following three: (1) a facility to require actors to enter comments when an object is moved backwards in the workflow (or more bluntly, rejected), (2) several additional reports

exposing action times, and (3) a mechanism to capture (in an auditable sense) the quality control checks performed on the object before its final delivery.

In the case of the subject system Steps 1, 2, and 3 took 18 months to bring the processes and tools to the current state where they have now been operating for an additional 18 months.

Step 4 - Analyze the Instrumentation Data

In preparation for building the Discrete Event Simulation (DES) models for Step 5 and Step 6, the data collected by the system is mapped to typical DES data sets, e.g., Inter-arrival Times (IATs), Processing times, etc. Depending on design decisions in Steps 2 and 3, this task may be straightforward or complicated. Figure 9 provides a pictorial representation of this step and its two immediate successors.

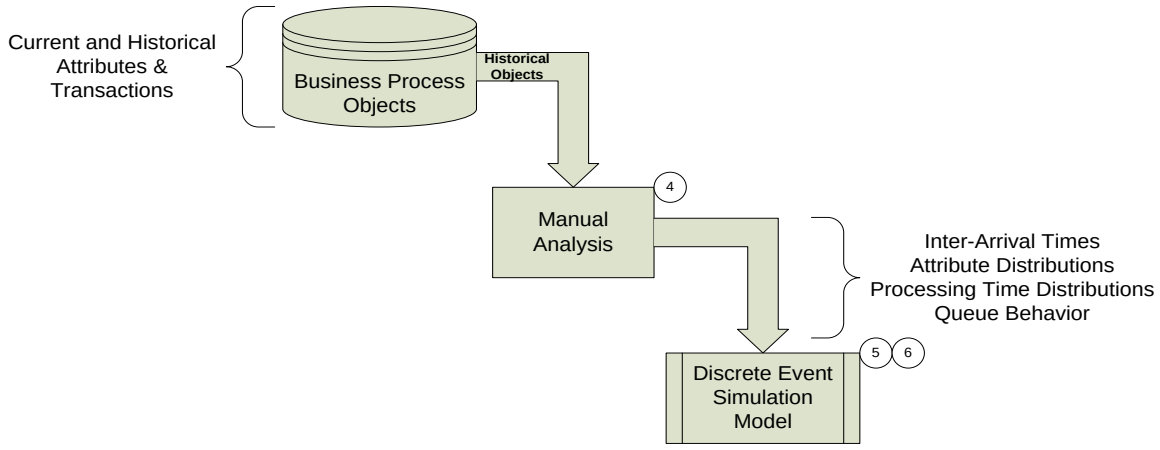


Figure 9 - Phase 2 of the system development

In the case of the test system, the raw IATs are trivial to extract from the transactional tables; however, extracting the net IATs (discounting nights, weekends and holidays) is more challenging since processing can take place at anytime but as a practical matter largely occurs between 7 AM and 7 PM in the Eastern US time zone and Monday through Friday. These net IATs are required to adequately create additional objects in the DES behind the object in question when there are multiple processing stations across multiple process steps such that jobs may overtake others during processing. The complexities of this relationship are described mathematically in Chapter Three.

Similarly, and as a consequence of "Practical Consideration #2" (APPENDIX B: PRACTICAL CONSIDERATIONS), basic workflow process events (when an object reaches a station's queue, and when it leaves the station) have to be decomposed to distinguish between queuing time and processing time. Since the actual start of processing is not captured and the queuing behavior is not necessarily First in First out (FIFO) or Last in First out (LIFO), but somewhat arbitrary, the decomposition requires a non-trivial approach and some effort to design and implement. The description of this portion of the process follows in Chapter Four.

As a note, the author has chosen to implement the analytical and modeling aspects of the subject system in Java. The reasons for this selection are more practical than theoretical as there are several existing DES frameworks written in Java, and the author is reasonably comfortable coding in Java. A practitioner could as easily choose to implement the analytical

and modeling aspects in another language as there is no elements of the solution that require Java or even an Object Oriented programming language.

These processing times are a critical input to the embedded modeling process that will generate a predicted delivery schedule for the object. Analysis of the data also aids in determining how jobs are handled at the various processing stations – an initial analysis of a sample of data for one step of the process indicated that jobs were being handled in a predominantly LIFO fashion for that step. The complete results of this analysis are portrayed in Figure 17, in Chapter Four.

The final activity in this step is to automate the analyses performed above so that the analyses can be orchestrated to run as required by the workflow system. To keep the development manageable, the author also coded these development tools in Java. The manual analysis of the target system consumed two weeks and the re-creation of the analytical process as an automated task took several more.

For steps in the process that exhibit readily identifiable queuing behavior, the analysis and automation will be more straightforward, however, the real-world nature of the process may lead to inconsistent behavior which would be considerably more challenging to model, especially in an automated sense.

Step 5 - Create a Discrete Event Simulation Model

With the Object-Actor-Action diagrams created in step 1 and validated by the end of step 3, and armed with IAT distributions, queue behaviors, and processing times extracted during step 4, a DES model can be readily encoded in a discrete event simulation tool (Arena, ProModel, etc.) for visualization, verification and validation. The only exception to this may be queuing behavior if the completed analysis from step 4 indicates non-standard behavior across the stations. In this case, it might be necessary to build modules for the DES framework to provide this behavior (see Chapter Three). After the model is built, operational validity will be established using historical data validation [9]. In this method, arrivals, processing times, and queuing behavior taken from the actual system will be used to stimulate the model. To determine this validity objectively, confidence intervals will be computed for both the historical and model generated cycle times, and these will be compared for statistically significant differences between the means [10].

It will be important to capture the entities and all of their attributes so that they may be fed into the embedded model from the upcoming step 6 to ensure the model behavior is consistent irrespective of the random variate seed behavior between the standalone DES environment and the Java-based DES framework. If the practitioner simply wants to conduct off-line simulations of the workflow process, he might stop here.

Step 6 - Create an Embedded Version of DES Model

In order for the DES model to be used to its greatest extent, it must remain synchronized with the production system. In this case re-coding the model, or alternatively building the initial model within a toolset that allows for stand-alone and embedded operation, is required. With the fully defined and validated behavior of the DES model from step 5 (as well as the full recording of its entities and attributes), the author coded a DES model using JSIM (one of several available Java DES frameworks) and re-ran the verification and validation with the recorded data from step 5 using the methodology previously described. At this point the Java-based, automated, analytical tools and the Java-based DES model will be ready for integration to the workflow system developed in step 2.

Step 7 - Integrate Model and Data Analysis Tools to Workflow Tool

As represented in Figure 10, the data required to update the machine learning process, pre-load the queues of the DES, and inform new object creation within the DES are all stored in the database that provides persistence for the workflow system. As a practical consequence, the integration of both the data analysis tools and the embedded DES model largely devolve to (1) connecting these items to the database, and (2) providing some mechanism to initiate their functions programmatically.

More specifically, the author will integrate the data analysis tools from step 4 with the transactional data from the workflow system to allow on-the-fly regeneration of the best machine learning model (described in Step 4) and provide updated IAT and attribute distributions. This portion of the process is required when a new object arrives, though it is not dependent on the object itself and so can be called without parameters. With an updated machine learning model the next task is to assign processing times to the newly arrived object based on its attributes. Since this task is clearly dependent on the new object a mechanism is required that refers the analysis to the object in question. The predicted processing times output from the machine learning model for the object will be stored with the object in the database. The penultimate task in the integration is to start the DES model with the current workflow system state loaded, the new object as the next arrival, and subsequent, synthetic objects created behind the object in question based on previous system behavior. To achieve this effect, the system need only call the DES model with the object in question being specified, and then only by reference. The final task is to output the predicted exit times for the object from each step in the business process to some level of prediction confidence. This output should be stored in the database with the object.

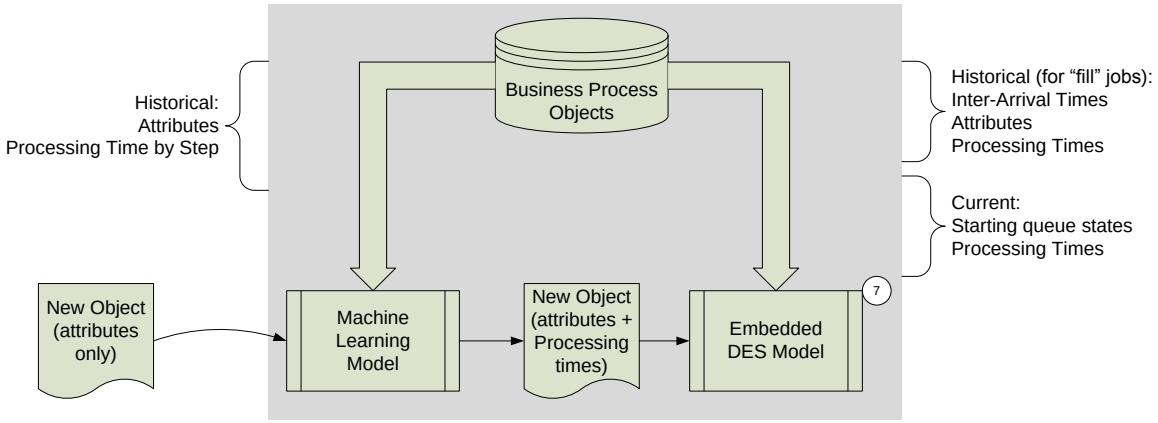


Figure 10 - Phase 3 of the system development

The details of executing steps 6 and 7 for the prototype are provided in Chapter Four.

Step 8 - Run Model in Non-Intrusive Mode

With the predictive subsystem integrated and tested in a development environment, what follows is the mundane migration of the prediction subsystem into the production environment. In a well designed and implemented development control system, this should require little more than the installation of the code on the production servers and modification of either an environment variable or initialization script to point at the production database instead of the development instance.

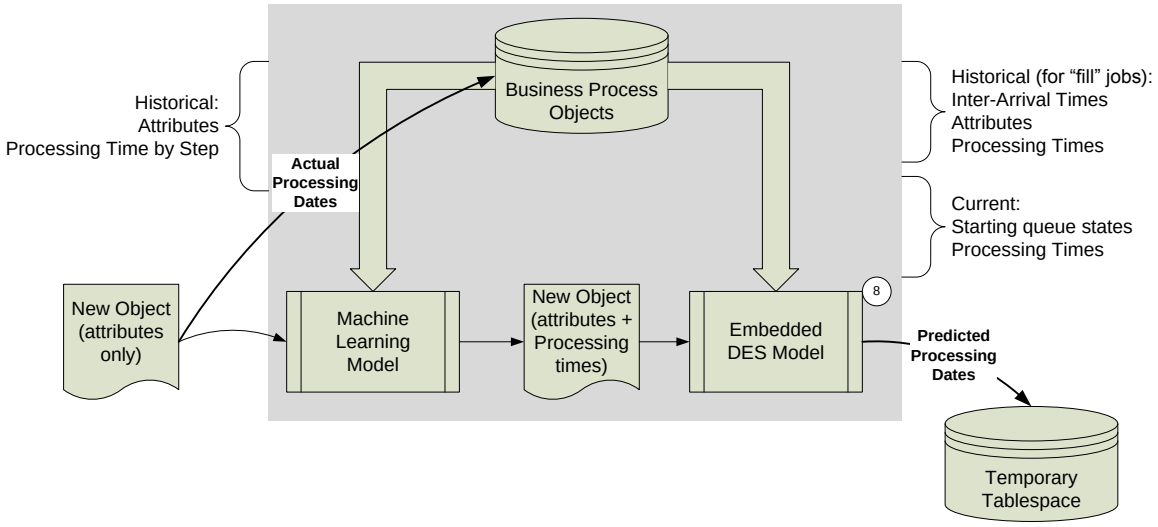


Figure 11 - Phase 4 of the system development

Until the quality of prediction versus actual performance has been validated, it is wise to keep the predictions out of view of the actors (as depicted in Figure 11) in the system (1) to avoid poor first impressions, and (2) to keep from skewing the results by providing intermediate target dates that are either too aggressive or too conservative (though this becomes an interesting capability to introduce in the final solution, aiming for an aggressive 80% confidence target while advertising to meet a conservative 90% confidence goal).

Step 9 - Validate Predictive Capabilities

After the predictive subsystem has been exercised in the production environment for a period of time, the actual intermediate and final dates for the objects processed in that time can be compared to the predicted dates generated by the prediction subsystem. Given the

throughput observed on the subject system, a period of 30 calendar days (22 business days) should provide approximately 100 new objects. From a practical standpoint the actual dates will be compared to the prediction intervals constructed for each object and will be deemed acceptable if, in fact, the actual dates fall within the intervals at the rate specified, e.g. 90 of 100 dates predicted fall within the 90% confidence prediction intervals.

Inadequacies at the individual step level, if discovered, may need to be addressed within the model (queuing behavior in particular) or within the analysis processes that build the machine learning model or output the processing times. Depending on the scope of the changes required to achieve acceptable performance, it may be necessary to return as far back as step 4 and cycle through some or all of the intervening Steps before re-executing step 8. The final results of steps 8 and 9 for the prototype are included in Chapter Five.

Step 10 - Activate Model for Process Scheduling

Once predictions match measured performance as described above, the workflow system will be reconfigured to publish the output of the prediction subsystem to the production scheduling table(s) as shown in Figure 12. With these promised delivery dates available in the system we can, with customer concurrence, switch our performance based metrics away from the existing gross measures of central tendency to measuring individual performance against discrete orders.

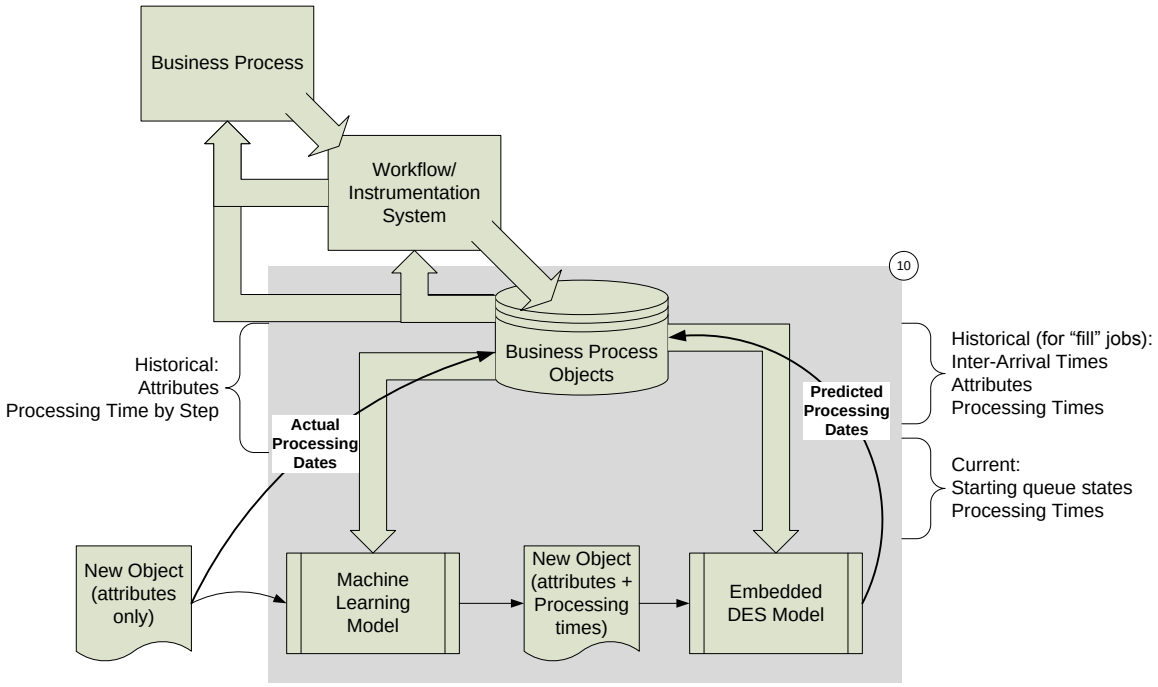


Figure 12 - Phase 5 of the system development

The following chapters will catalog the results of executing the 10-step process described in this section.

CHAPTER THREE: A HEURISTIC FOR DECOMPOSING TRANSACTION LOGS FROM WORKFLOW SYSTEMS

Introduction

The execution of Step 4 of the 10-step process described in Chapter Two brought to light the need for a non-deterministic method of decomposing the collection of transactions from the WFMS's logs into two vectors of observations – one representing the processing times for the jobs processed at a given station, and the other representing the queuing behavior of that station. This chapter describes the author's solution to this problem.

Formulation

To summarize the problem at-hand, consider the following formulation:

n_i: number of operations for job *i*

p_{ij}: processing time for job *i* at step *j* in its flow shop routing

w_{ij}: waiting time for job *i* at step *j*

f_{ij}: flow time for job *i* at step *j*, $f_{ij} = p_{ij} + w_{ij}$

f_i: flow time for job *i*

e_i: margin of error associated with job *i*, $e_i = d_i - \hat{d}_i$

r_i: release date for job *i*, i.e., the date that job *i* enters the WFMS

$\hat{d}_i$: quoted due date for job *i*, $\hat{d}_i = r_i + f_i + e_i$

Refactoring this formulation as shown in Equation 3.1 allows for segregation of data elements that are required for due date quoting based on the source and uncertainty of the data. The release date is given. The processing times are drawn for an appropriate distribution. The error may be assumed or estimated from historical performance, and the waiting times are related to the number of jobs in queue and queuing behavior.

$$\widehat{\boldsymbol{d}}_{\boldsymbol{i}} = \boldsymbol{r}_{\boldsymbol{i}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{p}_{\boldsymbol{ij}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij}} + \boldsymbol{e}_{\boldsymbol{i}} \quad (3.1)$$

Equation 3.2 summarizes the salient difficulty in predicting turn-around times (TATs) in a system with non-standard queuing behavior.

$$\boldsymbol{W}_{\boldsymbol{i}} \equiv \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij=}}\, \boldsymbol{f}\left(\boldsymbol{IAT}, \overrightarrow{\boldsymbol{P}}, \overrightarrow{\boldsymbol{Q}}, \overrightarrow{\boldsymbol{R}}\right). \quad (3.2)$$

Where IAT is the inter-arrival time for jobs that appear after the arrival of job *i*, and $\overrightarrow{P},\ \vec{Q}, \text{and } \vec{R}$ are the vectors of processing times, queuing behaviors, and rework rates respectively for the other jobs in the system. Note that the arrival process need not be stationary, and in fact, is not in the subject system [11].

Relevant Literature

There is generally a significant quantity of attribute data associated with the objects entering and flowing through a WFMS. van der Aalst, Reijers et al. make the point that modern information systems (and specifically workflow systems) capture much of the necessary data to perform data mining on the process information, which they termed “process mining” without

having to resort to external data collection though there have been few real-world exploitations of this capability captured in the literature [12]. Rozinat, Wynn et al. proposed to extend this concept through the use of a pair of open source tools -- YAWL (Yet Another Workflow Language) and ProM (Process Miner). They described the potentially tight coupling theoretically possible between a workflow system and a simulation model that represents that system. This coupling would be accomplished by describing the workflow system in YAWL, running the resultant workflow description through the YAWL runtime, and then developing plug-ins for ProM that would (1) allow it to ingest the system design and (2) interpret the transaction and state information. Rozinat successfully created an example of this coupling using a simple credit processing workflow. It is important to note Rozinat's conclusion -- that while the concept seems valid, the creation of a generalized process for achieving coupling was not yet obtainable [13]. In addition to the limitations imposed by the developmental nature of Rozinat's plug-ins for reading YAWL information into ProM, there are also limitations based on ProM itself in that there currently are not facilities to support the generalized queues that are necessary to support certain real-world processes such as the one under consideration.

Methodology

The author's proposed solution to determining W_i is then to (1) construct an embedded DES model, (2) determine the parameters for that model applicable at the point in time where job *i* enters the system, (3) determine the properties of job *i* necessary for representation

within the model, and (4) to repeatedly execute the model until an acceptable margin of error on predicting its time in system can be achieved. In order to effect this methodology, however, the vectors $\vec{P}$ and $\vec{Q}$ must be determined.

Assumptions

The proposed methodology is developed based on the following list of assumptions: (1) there is exactly one processor at each step, (2) there is no forced idle time at the processors at the steps, and (3) the resultant processing times for each step may be represented using a distribution function.

Queuing Behavior

The author's formulation for attacking $\vec{Q}$ from Equation 3.2 is, conceptually, similar to executing a discrete event simulation (DES) in reverse. When conducting a discrete event simulation, the release time for a job, the processing time for a job, and the queuing policy for a station are specified as inputs (either deterministically or stochastically), and the output for the job is the departure time from the station. The flow time f_{ij} for job *i* at station *j* (or cycle time) is the difference between the departure time and the release time (see Figure 13).

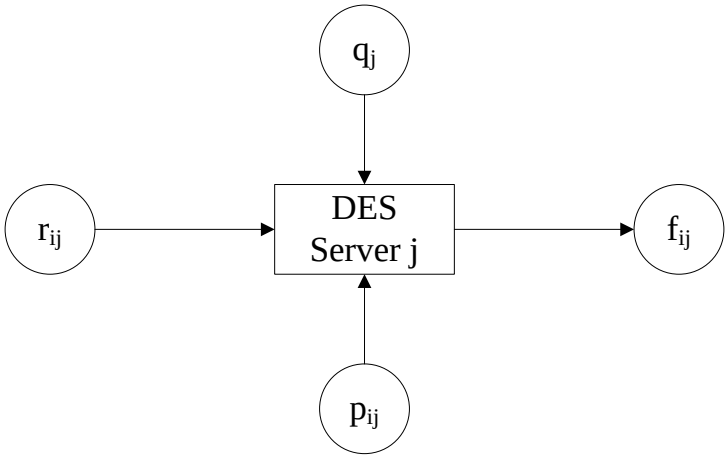


Figure 13 - Normal inputs and output from a DES Server

In the case where the transactional logs from the WFMS are given, however, the release and flow times are known and the result of the heuristic analysis are the processing time for the job, and the queuing behavior of the station (see Figure 14).

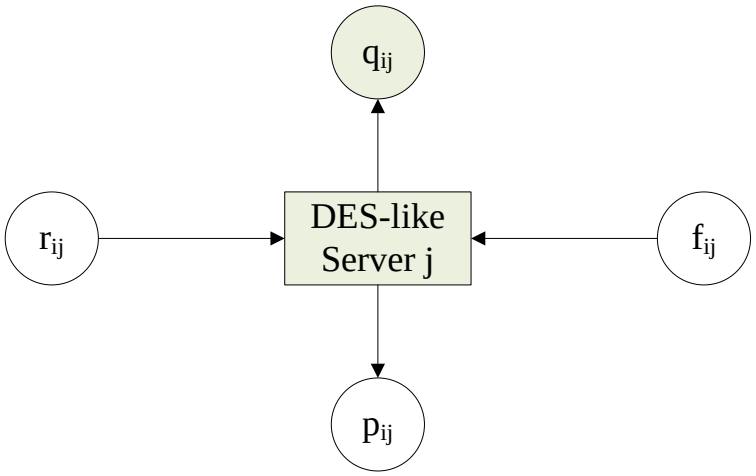


Figure 14 - Revised inputs and outputs available from virtual DES Server

More specifically, the historical jobs arriving at a given station are processed in time-order of their arrival at the station but the jobs are placed in the queue based on their recorded

departure time. Executing this process one job at a time, it is possible to determine the queue insertion location at the station, and the accumulated processing time for the job.

Heuristic Example

As an example of this process, consider the following sequence: Job 1, which arrives at Server *j* at time 0 and is known to have departed at time = 20, finds Server *j* empty and idle; since the server is empty and idle, Job 1 is immediately placed in service (location = 0, queue depth = 0) and begins to accumulate processing time. Job 2 (arrives at time = 5, will depart at time = 21) arrives at Server *j*; since the server is *not* idle the departure time of the newly arrived job is compared to that of the job in service; since Job 2 will depart after Job 1, it is placed in queue; since the queue is empty, Job 2 is queued at location = 1, queue depth =1. Job 3 (arrives at time = 10, will depart at time = 30) arrives at Server *j*; since Job1 is still in service, departure times for Jobs 1 and 3 are compared; Job 3 will depart after Job 1, so Job 3 is queued; since Job 3 will depart after Job 2, it is queued after Job 2 at location 2 and queue depth = 2. Job 4 (arrives at time = 15, will depart at time = 25) arrives at Server *j*; since its departure time is after Job 1 (still in service), Job 4 will be queued; since Job 4 will depart after Job 2 and before Job 3, it is queued at location = 2, queue depth = 3 which is recorded in as '2/3'. Executing this scenario, and stopping at time = 15 is represented graphically in Figure 15.

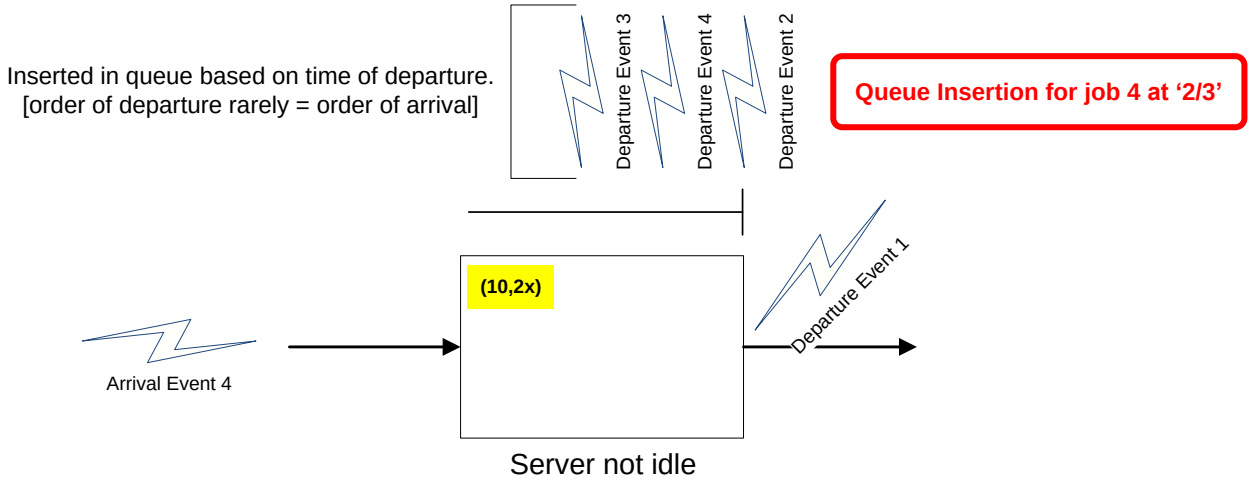


Figure 15 - Queue position determination

In pseudo-code, the virtual Server performs the following top-level tasks:

```
Read previous 180 days of Transactions for Server;
Create Arrival Events and Departure Events based on
       transactions for completed jobs;
loop through events in time order {
   if (arrival event) Push(event);
   else if (departure event) Pop(event);
}
```

The pseudo-code above references 180 days of transactions as the look-back window which is appropriate in the author's business environment. Depending on the circumstances of the practitioner's environment the look-back window might be appropriately specified in terms of days, or in terms of a number of transactions.

The virtual Server Push method performs the following:

```
new_job = get_job_from_event(event);
if (server idle)
 in_progress_job = new_job;
 new_job.arrival_location = 0;
 new_job.arrival_queue_depth = 0;
else (server busy)
 if (new_job.departure < in_progress_job.departure)
    in_progress_job.add_processing_time_to_date();
    queue.add(in_progress_job);
    in_progress_job = new_job;
    new_job.arrival_location = 0;
    new_job.arrival_queue_depth = queue.size();
 else
    queue.add(new_job);
    new_job.arrival_location = queue.find(new_job);
    new_job.arrival_queue_depth = queue.size();
```

The corresponding virtual Server Pop method performs the following:

```
in_progress_job.add_processing_time_to_date();
if (queue not empty)
 in_progress_job = queue.next();
else
 server idle = true;
```

The output of this function, which is accomplished by the “Push” method of the virtual server, is three parameters per station specifying the fraction of jobs that preempt, queue at the head-of-line, and queue at the tail-of-line. Jobs that do not meet any of the three criteria are assumed to be randomly placed in the queue between head-of-line and tail-of-line.

Processing Times

The second output of the process is the determination of the processing time for a job *i* at a particular Server *j*. And with these values in hand, the author can then fit the processing times with a statistical distribution. This statistical distribution addresses, in conjunction with the server simulation component, the $\vec{P}$ component from Equation 3.2.

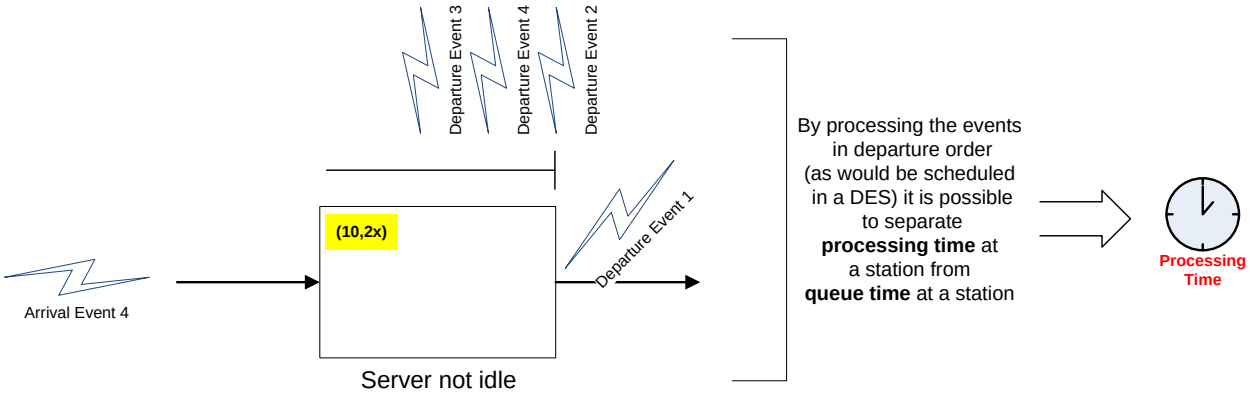


Figure 16 - Processing time determination

During the manual analysis process of step 4, the author used Rockwell Software's Input Analyzer (a component of their Arena product suite) to fit the processing time distributions and assess their "goodness of fit".

Summary

At the end of this process, the author faithfully captured the queuing behavior and processing time distributions which were then used as parameters in the stand-alone DES model for Step 5 which is described in Chapter Four. In Step 7, the entire analysis process

(extraction of queuing behavior, segregation of processing times, distribution fitting, goodness of fit testing, and time-based exponential smoothing) was automated through code written in Java. The use of this analytical process is described, along with the model output in Chapters Five and Six.

CHAPTER FOUR: PROCESSING PREDICTIONS THROUGH EMBEDDED SIMULATION

The following material was presented at the 2010 Software Engineering and Applications conference held by the International Association of Scientific and Technology for Development (IASTED), and published in the conference proceedings [11].

Abstract

Being faster is good. Being predictable is better. A faithful model of a system, loaded to reflect the actual system's state at a given point in time, can then be used to look into the future and predict performance. Building faithful models of processes with high degrees of uncertainty can be very challenging, especially where this uncertainty exists both in terms of processing times, and queuing behavior. The author will discuss the potential benefits of using a discrete event simulation to quote due-dates in a business process/work flow environment.

Introduction

In flush economic times the elements of excellence that characterize the practice of Industrial Engineering -- reducing cycle-times, decreasing variability, and increasing predictability can mean the difference between a growing business and a struggling one. In leaner times the consequences are more Boolean - the business survives, or it fails. The benefits of such pursuits are recognized. And these pursuits are common, though not ubiquitous in

manufacturing but appear much less frequently in human-centric business processes. Perhaps the reason for this discrepancy lies in the difficulty of capturing the seemingly capricious behavior of the humans in such a setting. In non-trivially complex business systems, the humans that perform functions within the business process do so with some measure of autonomy. This autonomy can lead to behavior, especially in the order that queued tasks are handled, that is difficult to capture and therefore to analyze. In this chapter the author asserts that a discrete event simulation (DES) model can be used to capture such behavior when augmented with a novel queuing component that allows for the flexible ordering of tasks within a queue.

Problem Formulation

To describe the situation mathematically, consider the following definitions and relationships:

n_i: number of operations for job *i*

p_{ij}: processing time for job *i* at step *j* in its flow

w_{ij}: waiting time for job *i* at step *j*

f_i: flow time for job *i*

e_i: margin of error associated with job *i*

l_i: lead time associated with job *i*

r_i: release date for job *i*, i.e. the date that job *i* enters the system

$\hat{d}_i$: quoted due date for job *i*

d_i: actual delivery date for job *i*

L_i: Lateness of job *i* with respect to its quoted due date

q: number of jobs in process or in queue when job *i* enters the system

Assuming that there is no down time at the steps and that there is no transportation time between steps, then the flow time for a job, f_i, is simply the sum of the expected processing times for the steps for that job, p_{ij}, and the expected waiting time per step for that job, w_{ij}.

$$\boldsymbol{f_i = \sum_{j=1}^{n_i}(p_{ij} + w_{ij})} \quad (4.1)$$

Then the lead time, l_i, used to quote a due date for that job is the flow time, f_i, plus some margin of error, e_i, associated with the estimation of the processing and waiting times.

$$\boldsymbol{l_i = f_i + e_i} \quad (4.2)$$

The predicted due date for the job, $\hat{d}_i$, is then the release date for the job into the system, r_i, plus the estimated lead time, l_i.

$$\boldsymbol{\hat{d}_i = r_i + l_i} \quad (4.3)$$

Refactoring this formulation as shown below allows for a more straightforward segregation of data elements that are required for due date quoting based on the source and uncertainty of the data. To wit: the release date is given, the processing times are drawn for an appropriate distribution, the error may be assumed or estimated from historical performance, and the waiting times are related to the jobs in queue and queuing behavior.

$$\widehat{\boldsymbol{d}}_{\boldsymbol{i}} = \boldsymbol{r}_{\boldsymbol{i}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{p}_{\boldsymbol{ij}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij}} + \boldsymbol{e}_{\boldsymbol{i}} \quad (4.4)$$

It is this relationship between the jobs in queue, the queuing behavior by job or by step, and the waiting times that can make this a challenging problem.

$$\boldsymbol{W}_{\boldsymbol{i}} \equiv \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij=}}\, \boldsymbol{f}\left(\boldsymbol{IAT}, \overrightarrow{\boldsymbol{P}}, \overrightarrow{\boldsymbol{Q}}, \overrightarrow{\boldsymbol{R}}\right) \quad (4.5)$$

Where IAT is the inter-arrival time for jobs that appear after job *i* arrives, and $\overrightarrow{P}, \vec{Q}, \vec{R}$ are the vectors of processing times, queuing behaviors, and re-work actions respectively for the other jobs in the system.

Completing the formulation, the lateness of a job, L_i, with respect to its quoted due date is simply the difference between the actual delivery date, d_i, and the quoted due date, $\hat{d}_i$.

$$\boldsymbol{L}_{\boldsymbol{i}} = \boldsymbol{d}_{\boldsymbol{i}} - \widehat{\boldsymbol{d}}_{\boldsymbol{i}} \quad (4.6)$$

The square of this lateness will be used as the measure of performance in the experiment described in Test Methodology section.

Related Literature

The following subsections summarize pertinent instances of the existing literature with respect to Due Date Quoting, and Business Process Modeling and Mining.

Due Date Quoting

Cheng and Gupta [14] produced a survey of the existing research with respect to due date determination. In this survey, Cheng and Gupta open by pointing out that meeting due dates is extremely important to practicing managers. They then utilize a classification scheme first proposed by Elion [15] which has six (6) dimensions: (1) Static versus Dynamic, (2) Deterministic versus Stochastic, (3) Single-product versus Multi-product, (4) Single-processor versus Multi-processor, (5) Theoretical versus Practical, and (6) Exogenous due-dates versus Endogenous due-dates. Since exogenous due-dates obviate due-date quoting and lead directly to sequencing and scheduling problems, Cheng and Gupta focus their attention on endogenous due-dates. Using the above classification scheme they conclude that there is very little extant research on Dynamic, Complex, Multi-processor systems. And after noting that better predictors would be beneficial, if practical, they conclude that there is a need for more practical and applied research in this area.

Alfieri [16] proposes two new quoting policies based on setting a static Safety Time (ST) parameter analogous to e_i in the formulation above noting that setting this parameter dynamically could be time consuming. The performance of these quoting policies, which both presuppose a First-Come-First-Served (FCFS) ordering, is compared to the Total Work Content (TWK) policy when jobs are sequenced by Shortest Processing Time (SPT), Earliest Due Date (EDD) and First-In-First-Out (FIFO). These comparisons are predicated on batch scheduling (ignoring subsequent arrivals), deterministic processing times and non-permutation

sequencing. With these simplifications, her results indicate that TWK outperforms both of her proposed policies. She notes that estimating flow times for more complicated systems is a suitable topic for future research.

Subsequent to the survey conducted with Gupta discussed above, Cheng [17] describes an efficient and optimal sequencing algorithm when using the slack due-date quoting policy. Cheng simplifies the system under consideration by assuming that once a set of jobs is sequenced, no subsequent jobs will affect the systems performance, there will be no re-sequencing of the jobs between stations and all of the earliness and tardiness costs are constant. In effect, the lack of consideration of arrivals and non-permutation scheduling becomes a presupposition of FCFS. In this scenario Cheng concludes that an SPT sequence is optimal although this conclusion is at odds with the findings of Duenyas and Hopp below.

Duenyas and Hopp [18] propose an analytical framework for evaluation of various job sequencing rules given that flow times can be optimally predicted. Working through a series of increasingly more generalized scenarios they conclude that an EDD sequence is optimal if the tardiness penalty is constant for all customers and proportional to the tardiness which seems to contradict Cheng [17] above. To achieve this result Duenyas and Hopp only assume that pre-emption does not take place. The result of an EDD sequence being optimal is useful in that it provides direction for redesigning the workflow system in this author's construct to encourage EDD processing order but is not helpful in determining the optimal due-dates.

Similar to Duenyas and Hopp above, Lawrence [19] presupposes that the practitioner either has a simple system with closed-form flow time estimates, or has some way to determine flow times for complex systems. With that as a precondition, he describes an analytical approach to setting due-dates based on previously observed forecasting errors. While Lawrence proposes to fit the forecasting errors, which he refers to as "G", using a Ramberg-Schmeiser distribution, he concludes that Erlang and Gaussian distributions worked equally well in his research. Lawrence makes three observations that are particularly germane in this context: (1) exponential smoothing of the forecasting error distribution parameters enhances the accuracy of the fit, especially in time-dynamic situations, (2) various measures of performance lead to differing uses of the error distribution, e.g. Mean Absolute Lateness is minimized by adding the median of the error distribution to the predicted flow time, Mean Square Lateness (MSL) is minimized by adding the mean of the distribution to the predicted flow time, and service level matching is met by adding the target percentile of the distribution to the predicted flow time, e.g. $G^{-1}(0.9)$ for a 90% Service Level, and (3) the analytic due date quoting policies that include information about the current system state outperform those that do not, at least in the simple scenarios that the author specifically evaluates. Additionally, Lawrence's paper provides a good summary of the most common analytic quoting policies which will be useful for comparison with this author's proposed modeling-based approach.

Van Ooijen and Bertrand [20] introduce a distinction in terminology intended to allow some leeway between the tightly estimated Internal Due Date (IDD) and the slightly looser

External Due Date (XDD). To set this difference, which is analogous to e_i in the problem description above, or the Safety Time from Alfieri, or Lawrence's error distribution, G, the authors propose to adjust the XDD using the ratio of the current level of work in progress (acwip) to the average level of work in progress (nwip). Using variations of this quoting policy various sequencing rules were applied and the optimal cost per order was established over a variety of relative earliness/tardiness combinations. Van Ooijen and Bertrand's results bring some closure to the disagreement between Cheng [17] and Duenyas [18] by noting that when earliness and lateness penalties are of similar magnitude then SPT sequencing works best; however, when tardiness penalties are much larger than earliness costs a Due Date sequencing rule is best. Another interesting conclusion that can be drawn from the data is that in spite of the dependence on FCFS sequencing in much of the literature, FCFS provided among the worst performance of the sequencing rules tested.

Rajasekera, Murr, et al [21] open by observing that including more information into the dynamic flow time prediction process produces better results. Much of the paper subsequently focuses on an analytical description of a load-balancing algorithm that could be implemented in an information system integrated with the manufacturing system. The authors conclude that after applying their load balancing procedure and assuming FCFS processing, then setting due-dates is straightforward even when taking into account the jobs already in the system. As a parting note, the authors concede that more complex work centers would require more complex queuing decomposition methods and further analysis.

Business Process Modeling and Mining

van der Aalst, Reijers et al. make the excellent point that modern information systems (and specifically workflow systems) capture much of the necessary data to perform data mining on the process information, which they term “process mining” without having to resort to external data collection though there have been few real-world exploitations of this capability captured in the literature [12].

Rozinat, Wynn et al. propose to extend the preceding concept through the use of a pair of open source tools -- YAWL (Yet Another Workflow Language) and ProM (Process Miner). They describe the potentially tight coupling theoretically possible between a workflow system and a simulation model that represents that system. This coupling would be accomplished by describing the workflow system in YAWL, running the resultant workflow description through the YAWL runtime, and then developing plug-ins for ProM that would (1) allow it to ingest the system design and (2) interpret the transaction and state information. Rozinat successfully created an example of this coupling using a simple credit processing workflow. It is important to note Rozinat’s conclusion that while the concept seems valid, the creation of a generalized process for achieving coupling was not yet obtainable [13]. In addition to the limitations imposed by the developmental nature of Rozinat’s plug-ins for reading YAWL information into ProM, there are also limitations based on ProM itself in that there currently are not facilities to support the generalized queues that are necessary to represent certain real-world processes such as the one under consideration.

Given the ongoing difficulties in creating an automated method of utilizing the workflow output logs to build a model of the system, this author is left with little choice but to build a discrete event simulation model of his system by hand.

Necessity Of A Novel Approach

As mentioned in the introduction, the author asserts that better predictive performance in quoting due dates should be achieved by making a faithful model of the system into which a new job is then introduced. The motivation for doing so, as well as the argument to support this assertion follows in two parts: Modeling versus deterministic assessment and Real-world versus ideal queuing behavior.

Necessity of Modeling

Meeting promised due dates is critical to customer satisfaction [14, 18, 19, 21].

Promised due dates are readily met when arbitrarily long lead times are set. However, quoting arbitrarily long lead times to ensure service levels dilutes customer appeal while overly optimistic lead times erodes customer confidence [16]. Based on this, more accurate due dates (with narrower confidence intervals) are better (more pleasing to customers) as long as the mechanism is practical to implement [14].

As expressed in the Problem Formulation section, the due-date for a job is dependent on that job's processing times and waiting times, and should also include some safety margin [16, 17, 19].

Also from the Problem Formulation section, the dominant feature of the due-date setting problem is estimating the wait time for a given job [14].

The wait times for a job are obviously dependent on the jobs already in the system, though the particular relationship is also dependent on the queuing scheme assumed [16, 18, 19].

Including more information about the current state of the system leads to better predictions of due dates [14, 16, 18-21].

Analytical methods are suitable for simple cases with ideal assumptions, but more complicated systems require more complicated analysis typically involving simulation [14, 16, 18].

A detailed discrete event simulation model of the actual system will allow more information on the system (design, historical performance, and current state) to be brought to bear on the estimation of waiting times.

Necessity of Real-world Queuing Behavior

The data observed from the subject system for this author's research exhibits job insertion at head of line preemptively, head of line without preemption, tail of line, and other locations in the middle of the queue as depicted in Figure 17.

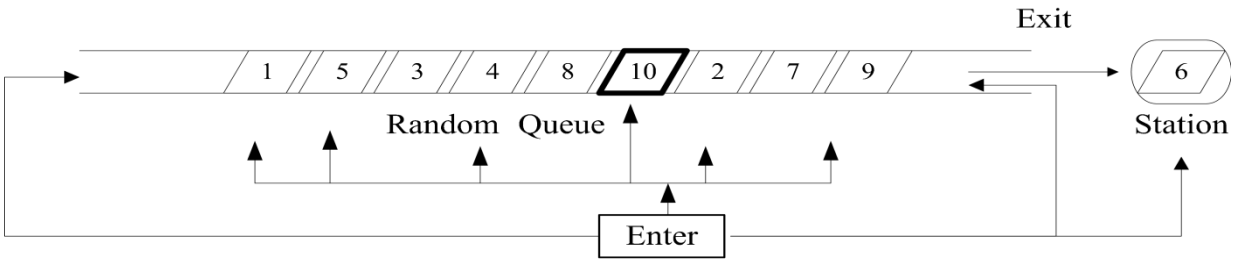


Figure 17 - Flexible Queue

Since the insertion location for a given job determines the minimum number of jobs that will be processed before that job, it provides a lower bound for the wait time of the target job at that step, but this determination is not complete, as subsequent jobs may arrive after the job in question and be queued in front of the target job increasing its wait time at that step.

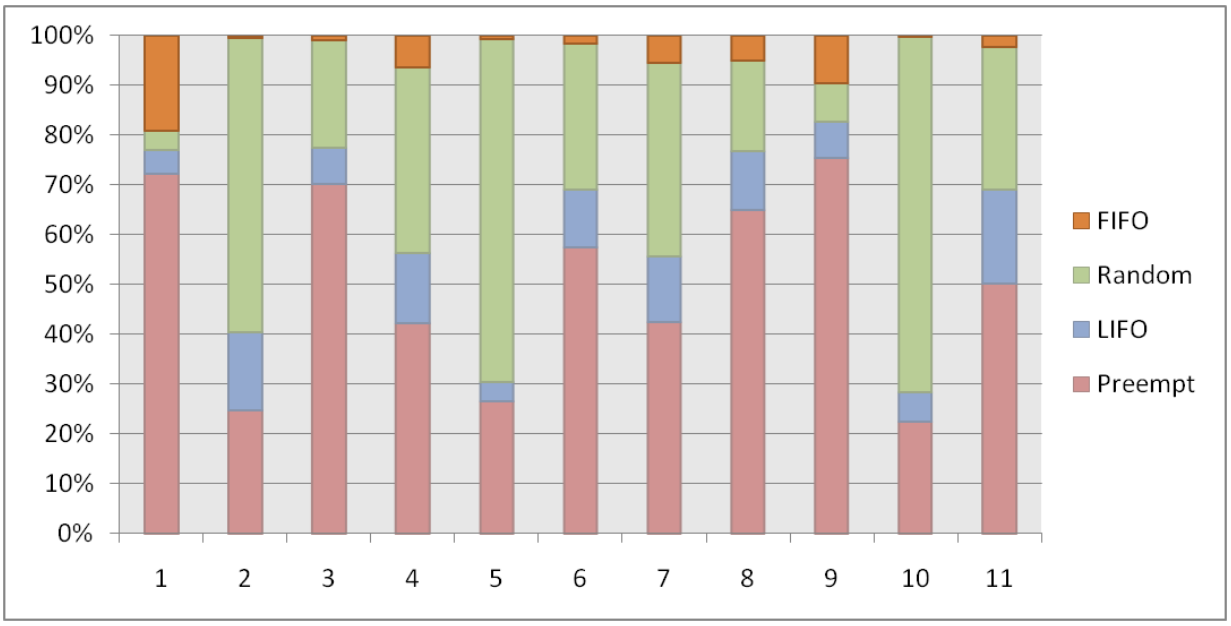


Figure 18 - Relative percentage of jobs inserted into queues by position

As mentioned in the Problem Formulation section, several thousand historical transactions are available for analysis of the system under test. By decomposing the transactions into corresponding arrival and departure events and then processing those events in departure order it is possible to glean the relative insertion position of jobs at each step. The results of this analysis are applied to the model of the system under test for this paper and expressed as the relative frequency of job insertion location by step as shown in Figure 18. These relative frequencies will be used in the empirical queuing implementation described in the System Under Test section. While all of the existing queuing models provide equivalent, average, system-level performance prediction, the author's goal is to accurately model the

behavior of a single, discrete job within the context of its fellow jobs, and therefore a more flexible model is required.

Argument Summation

In summary, more accurate assignment of due dates will make customers more likely to continue to place their orders using the system. Outside of certain idealized systems, incorporating more detail in the prediction process can make those predictions more accurate. A DES model allows for incorporating more system detail than any of the existing mechanisms and incorporating real-world queuing behavior is a key aspect of that mechanism. It is therefore worthwhile to study the forecasting performance of a faithful DES model against existing, deterministic policies.

System Under Test

The actual system that this example is based upon is a workflow system that supports a business process. It is similar to a flexible flow shop in which the stations that a job passes through are known (11 in this example) and the jobs in the stations' queues at any point are known. All of the other parameters associated with the job shop, including job processing times per station and station queuing behavior are uncertain though there is a significant body of past performance data that is brought to bear to determine input distributions.

This system was modeled in DES form using Rockwell's Arena package and an overview of the resultant model is depicted in Figure 19. A source module was instantiated which implements a Poisson arrival process for new orders and is labeled "New Orders" in Figure 19.

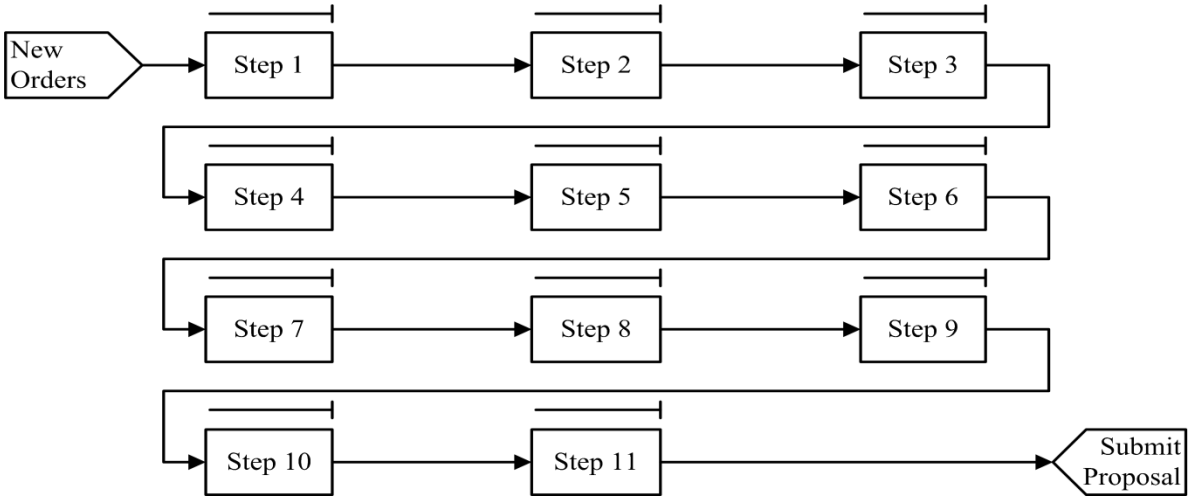


Figure 19 - Model of system

After the orders arrive in the system they are assigned processing times and queue behaviors using an assignment module based on the distributions as listed in Table 1 and Table 2 respectively. These values are stored in attributes associated with each order. Eleven servers were then instantiated, labeled "Step 1" through "Step 11", and connected serially as depicted.

The processing times for each order and at each server are read from the attributes assigned above. Associated with each server is a queue that can be configured to process orders as FIFO, Last In-First Out (LIFO), or in priority order based on an assigned attribute. The model is completed by instantiating an order sink which disposes of the orders after processing is complete – this component is labeled "Submit Proposal" in Figure 19. To capture the actual

departure dates from the system and aid with the experiment a series of output modules (not shown) are instantiated. These modules allow for the capture of the squared lateness by job with respect to each of the due date quoting policies previously mentioned.

As mentioned in the Necessity of a Novel Approach section, transactions from the actual workflow system were decomposed into arrival and departure events. In addition to providing data for queuing behavior, this event processing also partitioned the time each job spent at a server into processing time and waiting time. Using Rockwell's Input Analyzer, the processing time data was fitted. The outputs of this process are the following processing time distributions as listed in Table 1.

Table 1 - Processing times by step

Step	Processing Time Distribution
Step 1	WEIB(0.146, 0.389)
Step 2	WEIB(1.19, 0.425)
Step 3	WEIB(0.404, 0.304)
Step 4	WEIB(0.709, 0.407)
Step 5	WEIB(0.928, 0.417)
Step 6	WEIB(0.573, 0.342)
Step 7	WEIB(0.821, 0.386)
Step 8	WEIB(0.505, 0.34)
Step 9	WEIB(0.373, 0.331)
Step 10	WEIB(0.918, 0.405)
Step 11	WEIB(1.32, 0.463)

Similarly, the following Queuing Distributions (see Table 2) were also fitted using Input Analyzer based upon the previously described convolution of the historical data such that the input position is mapped to fall between 0 for head of line and 1 for tail of line.

Table 2 - Queuing behavior by step

Step	Processing Time Distribution
Step 1	WEIB(0.00947, 0.33)
Step 2	BETA(0.413, 1.49)
Step 3	WEIB(0.00474, 0.399)
Step 4	LOGN(2.37, 187)
Step 5	BETA(0.355, 0.86)
Step 6	WEIB(0.0257, 0.373)
Step 7	WEIB(0.0978, 0.415)
Step 8	WEIB(0.0276, 0.328)
Step 9	WEIB(0.0119, 0.33)
Step 10	BETA(0.401, 1.14)
Step 11	LOGN(1.1, 70.7)

As a detail of the implementation, both the Processing Times and Queuing Behavior distributions were assigned unique random variate streams (avoiding Arena's default stream of 10). Note that the Queuing Behavior distributions only affect the model when the Queue Mode is set to prioritize the queue by lowest attribute value.

Additional entity attributes were defined and assigned in an Arena "Assignment" module to capture the calculated due dates based on the JIQ (Jobs In Queue), SLK (Slack assignment), NOP (Number of Operations), and TWK (Total Work Content) policies as described by Cheng and Gupta [14]. These policies are represented by the following formulae:

JIQ: $\hat{d}_i = r_i + JIQK1 * \sum_{j=1}^{n_i} p_{ij} + JIQK2 * q$ (4.7)

SLK: $\hat{d}_i = r_i + SLKK + \sum_{j=1}^{n_i} p_{ij}$ (4.8)

NOP: $\hat{d}_i = r_i + NOPK * n_i$ (4.9)

TWK: $\hat{d}_i = r_i + TWKK * \sum_{j=1}^{n_i} p_{ij}$ (4.10)

Note that each of the policies has one or more coefficients (JIQK1, JIQK2, SLKK, NOPK, TWKK) which must be adjusted based on the actual model. The due dates captured in these attributes were then used to calculate the Squared Lateness of the entities by due-date quoting policy and then recorded as outputs of the model.

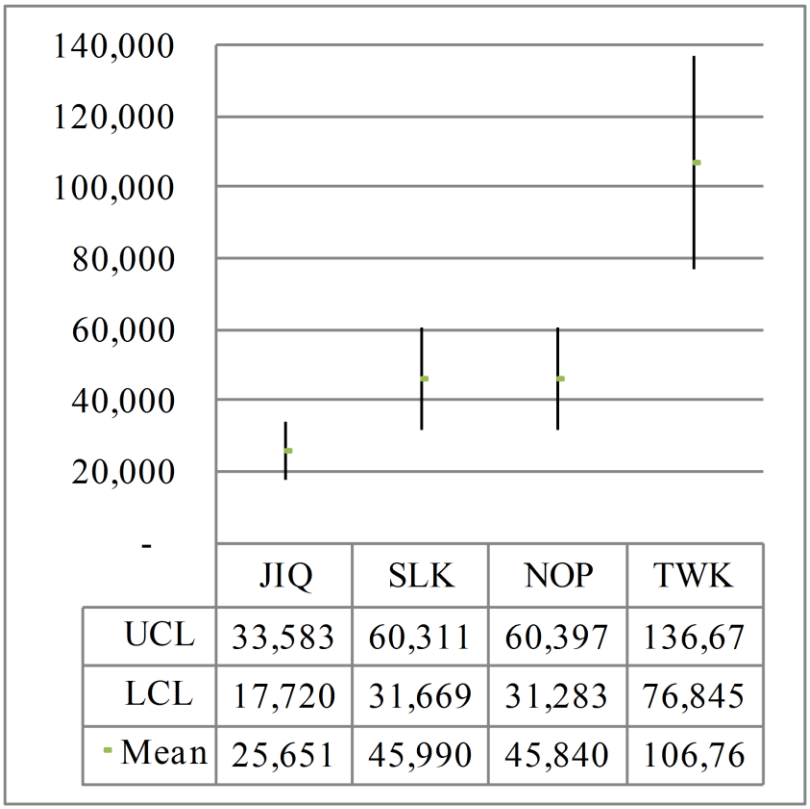
Test Methodology

Before comparison of the due-date quoting policies could be undertaken, the adjusting parameters for each of the policies had to be tuned. Rockwell's Process Analyzer was used to adjust the coefficients for each policy (JIQK1, JIQK2, SLKK, NOPK, TWKK) while minimizing its Mean Square Lateness performance. After these coefficients were tuned, the model was set up to run with a 90 day warm up and 365 days of simulation in each of three queuing modes: FIFO, LIFO, and Empirical. It is in this last mode that the Queuing Behavior attributes (listed in Table 2) come into play by prioritizing the entities by the value drawn from that distribution for that step.

The model was executed for 30 replications in each mode and the output captured. Since the Mean Square Lateness was recorded as an "Output" Arena politely exports the mean and 95% confidence half-widths directly in the output file.

Test Results

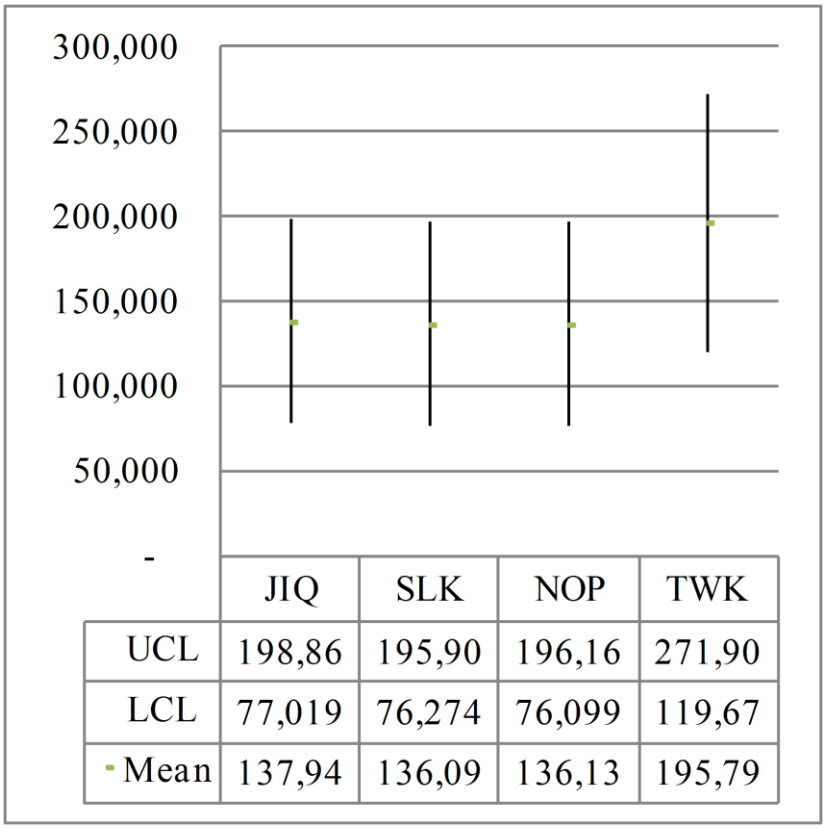
The following three figures display the relative performance of the four due date quoting policies that were tested using this model. Given the assumptions taken when these policies were developed, it is not surprising that the results of the first test case align reasonably with that summarized from the literature under the section titled Due Date Quoting as shown in Figure 20 below using Microsoft Excel's High-Low-Close Stock chart to handily portray the confidence interval for the MSL per policy.



	JIQ	SLK	NOP	TWK
UCL	33,583	60,311	60,397	136,67
LCL	17,720	31,669	31,283	76,845
Mean	25,651	45,990	45,840	106,76

Figure 20 - Mean Squared Lateness – FIFO

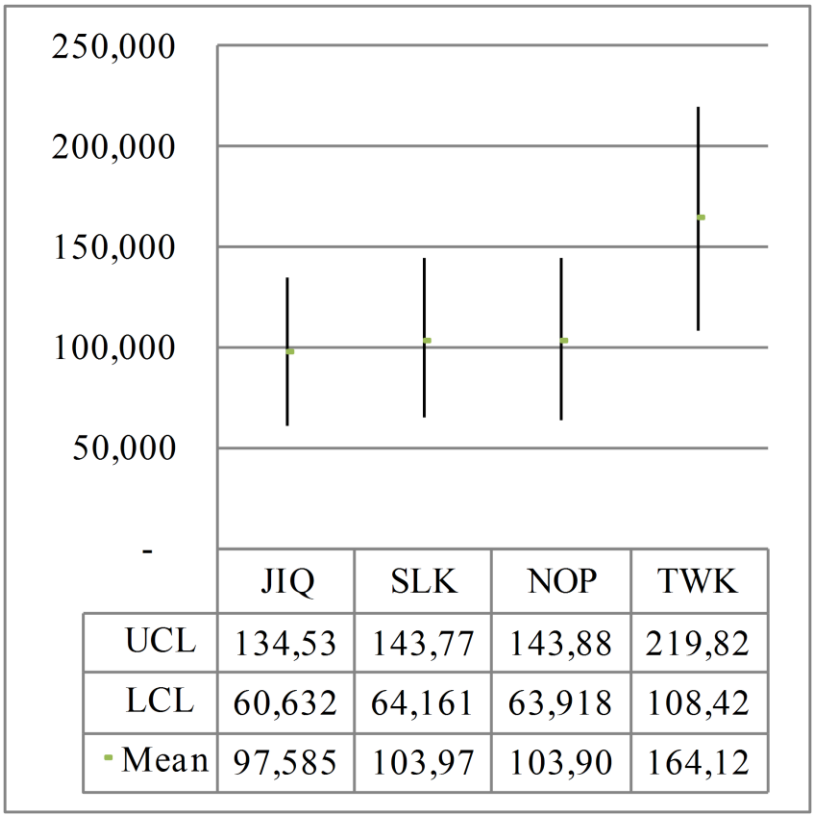
Similarly, Figure 21 shows the relative performance of the policies when the queuing behavior is switched for FIFO to LIFO. As argued above in the section titled Necessity of a Novel Approach, this drastic reduction in performance when the system does not conform to the simplifying assumptions is not surprising. It is worth noting that not only does the performance suffer greatly, but that the variance in the squared lateness is large enough that the policies are no longer distinguishable statistically.



	JIQ	SLK	NOP	TWK
UCL	198,86	195,90	196,16	271,90
LCL	77,019	76,274	76,099	119,67
Mean	137,94	136,09	136,13	195,79

Figure 21 - Mean Squared Lateness - LIFO

And finally, in Figure 22, the corresponding results are portrayed when the queuing mode incorporates the fitted distributions from Table 2. Given that the distributions indicate behavior between FIFO and LIFO in an approximate 40%/60% split the results below are between the two previous results sets.



	JIQ	SLK	NOP	TWK
UCL	134,53	143,77	143,88	219,82
LCL	60,632	64,161	63,918	108,42
Mean	97,585	103,97	103,90	164,12

Figure 22 - Mean Squared Lateness - Empirical

The broad confidence intervals of the latter two test cases dictate larger than reasonable margins required to meet desired service levels.

One of the test cases that the author had intended to address was the addition of pre-emption for head of line insertions. Unfortunately, Arena does not readily support pre-emptive processing with its built-in queuing component.

Conclusion And Future Research

All of the tested due-date quoting policies tested suffered when applied to systems that did not inherently provide FCFS behavior. Clearly there is room for additional research on setting due dates in non-FCFS systems such as those that are prevalent in more human-centric systems. As borne out by the test results, in such situations the relationship between the jobs in queue, the queuing behavior, and the wait times for the orders is too complex to be adequately captured by the prevalent due date quoting policies and should benefit from the computational flexibility provided by a discrete event simulation.

An additional source of complexity in the production system could be represented in a DES model by the inclusion of a 3-way decision block that represents the likelihood that a given job will be accepted (and thus passed to the next step), rejected (and returned to the previous step), or returned to the customer with no further action – this behavior was omitted from the model used in this experiment but is implemented in the embedded models described in Chapters Five, Six and Seven.

The author has also created and incorporated a more robust queuing component that will support random queue placement as well as pre-emption for use in embedded DES simulations and the prototype system described in the following chapters.

CHAPTER FIVE: PREDICTING BUSINESS PROCESS PERFORMANCE WITH 'REAL WORLD' QUEUING

The following material has been submitted for presentation at the 2011 Interservice/Industry Training, Simulation and Education Conference.

Introduction

Accurate determination of due dates for the delivery of bespoke items based on non-technical specifications is a challenging task. Limiting fixed staffing levels to control costs is at odds with having sufficient resources necessary to quote these due dates in a timely fashion. An environment that is extremely contentious with respect to the necessary resources and offering little in the way of firm prioritization only exacerbates the situation. And finally, when customers demand both demonstrably strict dates and penalties for exceeding those dates the situation becomes nearly untenable. The author proposes that an artful combination of automated analysis and efficient simulation might be successful in resolving this stark situation.

Prerequisites

In order to apply the methodology described here, a practitioner should already have (1) developed a functional, transaction-based workflow system, (2) performed an initial, manual data analysis of the processing times, queuing behavior and rework rates, and (3) built a representative discrete event simulation (DES) model of the workflow process to validate

understanding of the practitioner's system. In this author's case, the model of the system at hand is depicted in Figure 23.

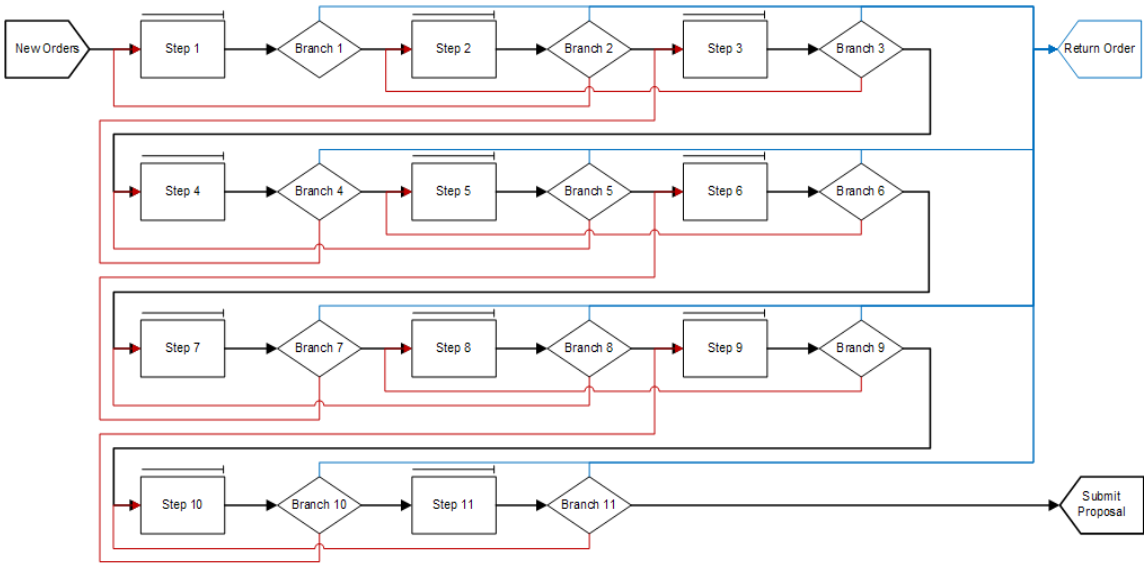


Figure 23 - Detailed DES model of system

Scope of Problem

To summarize the problem at-hand, consider the following abbreviated formulation:

n_i: number of operations for job *i*

p_{ij}: processing time for job *i* at step *j* in its flow

w_{ij}: waiting time for job *i* at step *j*

e_i: margin of error associated with job *i*

r_i: release date for job *i*, i.e. the date that job *i* enters the system

$\hat{d}_i$: quoted due date for job *i*

Refactoring this formulation as shown below allows for a more straightforward segregation of data elements that are required for due date quoting based on the source and uncertainty of the data. To wit: the release date is given, the processing times are drawn for an appropriate distribution, the error may be assumed or estimated from historical performance, and the waiting times are related to the number of jobs in queue and queuing behavior.

$$\widehat{\boldsymbol{d}}_{\boldsymbol{i}} = \boldsymbol{r}_{\boldsymbol{i}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{p}_{\boldsymbol{ij}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij}} + \boldsymbol{e}_{\boldsymbol{i}} \,. \tag{5.1}$$

The following relationship summarizes the salient difficulty in predicting turn-around times (TATs) in a system with non-standard queuing behavior.

$$\boldsymbol{W}_{\boldsymbol{i}} \equiv \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij=}}\, \boldsymbol{f}\left(\boldsymbol{IAT}, \vec{\boldsymbol{P}}, \vec{\boldsymbol{Q}}, \vec{\boldsymbol{R}}\right) . \tag{5.2}$$

Where IAT is the inter-arrival time for jobs that appear after job *i* arrives, and $\vec{\mathrm{P}},\ \vec{\mathrm{Q}}, \text{and } \vec{\mathrm{R}}$ are the vectors of processing times, queuing behaviors, and rework rates respectively for the other jobs in the system. Note that the arrival process need not be stationary, and in fact, is not in the subject system [11].

The author's proposed solution to determining W_i is then to (1) construct an embedded DES model, (2) determine the parameters for that model applicable at the point in time where job *i* enters the system, (3) determine the properties of job *i* necessary for representation within the model, and (4) to repeatedly execute the model until an acceptable margin of error on predicting its time in system can be achieved.

Relevant Literature

The following sections will highlight some of the salient literature that bears upon this topic from the areas of due date quoting, predictive use of models, and embedded modeling.

Due Date Quoting

Cheng and Gupta [14] produced a survey of the existing research with respect to due date determination. In this survey, Cheng and Gupta open by pointing out that meeting due dates is extremely important to practicing managers. They then utilize a classification scheme first proposed by Elion [15] which has six (6) dimensions: (1) Static versus Dynamic, (2) Deterministic versus Stochastic, (3) Single-product versus Multi-product, (4) Single-processor versus Multi-processor, (5) Theoretical versus Practical, and (6) Exogenous due dates versus Endogenous due dates. Since exogenous due-dates obviate due-date quoting and lead directly to sequencing and scheduling problems, Cheng and Gupta focus their attention on endogenous due-dates. Using the above classification scheme they conclude that there is very little extant research on Dynamic, Complex, Multi-processor systems. And after noting that better predictors would be beneficial, if practical, they conclude that there is a need for more practical and applied research in this area.

Alfieri [16] proposes two new quoting policies based on setting a static Safety Time (ST) parameter analogous to e_i in the formulation from Chapter Three noting that setting this

parameter dynamically could be time consuming. The performance of these quoting policies, which both presuppose a First-Come-First-Served (FCFS) ordering, is compared to the Total Work Content (TWK) policy when jobs are sequenced by Shortest Processing Time (SPT), Earliest Due Date (EDD) and First-In-First-Out (FIFO). These comparisons are predicated on batch scheduling (ignoring subsequent arrivals), deterministic processing times and non-permutation sequencing. With these simplifications, her results indicate that TWK outperforms both of her proposed policies. She notes that estimating flow times for more complicated systems is a suitable topic for future research.

Subsequent to the survey conducted with Gupta discussed above, Cheng [17] describes an efficient and optimal sequencing algorithm when using the slack due date quoting policy. Cheng simplifies the system under consideration by assuming that once a set of jobs is sequenced, no subsequent jobs will affect the systems performance; there will be no re-sequencing of the jobs between stations and all of the earliness and tardiness costs are constant. In effect, the lack of consideration of arrivals and non-permutation scheduling becomes a presupposition of FCFS. In this scenario Cheng concludes that an SPT sequence is optimal although this conclusion is at odds with the findings of Duenyas and Hopp below.

Duenyas and Hopp [18] propose an analytical framework for evaluation of various job sequencing rules given that flow times can be optimally predicted. Working through a series of increasingly generalized scenarios they conclude that an EDD sequence is optimal if the tardiness penalty is constant for all customers and proportional to the tardiness which seems to

contradict Cheng [17] above. To achieve this result Duenyas and Hopp only assume that preemption does not take place. The result of an EDD sequence being optimal is useful in that it provides direction for redesigning the workflow system in this author's construct to encourage EDD processing order but is not helpful in determining the optimal due dates.

Similar to Duenyas and Hopp above, Lawrence [19] presupposes that the practitioner either has a simple system with closed-form flow time estimates, or has some way to determine flow times for complex systems. With that as a precondition, he describes an analytical approach to setting due dates based on previously observed forecasting errors. While Lawrence proposes to fit the forecasting errors, which he refers to as "G", using a Ramberg-Schmeiser distribution, he concludes that Erlang and Gaussian distributions worked equally well in his research. Lawrence makes three observations that are particularly germane in this context: (1) exponential smoothing of the forecasting error distribution parameters enhances the accuracy of the fit, especially in time-dynamic situations, (2) various measures of performance lead to differing uses of the error distribution, e.g. Mean Absolute Lateness is minimized by adding the median of the error distribution to the predicted flow time, Mean Square Lateness is minimized by adding the mean of the distribution to the predicted flow time , and service level matching is met by adding the target percentile of the distribution to the predicted flow time, e.g. $G^{-1}(0.9)$ for a 90% Service Level, and (3) the analytic due date quoting policies that include information about the current system state outperform those that do not at least in the simple scenarios that the author evaluates specifically. Additionally, Lawrence's

paper provides a good summary of the most common analytic quoting policies which will be useful for comparison with this author's proposed modeling-based approach.

Van Ooijen and Bertrand [20] introduce a distinction in terminology intended to allow some leeway between the tightly estimated Internal Due Date (IDD) and the slightly looser External Due Date (XDD). To set this difference, which is analogous to e_i in the problem description from section 1.2, or the Safety Time from Alfieri, or Lawrence's error distribution, G, the authors propose to adjust the XDD using the ratio of the current level of work in progress (acwip) to the average level of work in progress (nwip). Using variations of this quoting policy various sequencing rules were applied and the optimal cost per order was established over a variety of relative earliness/tardiness combinations. Van Ooijen and Bertrand's results bring some closure to the disagreement between Cheng [17] and Duenyas [18] by noting that when earliness and lateness penalties are of similar magnitude then SPT sequencing works best; however, when tardiness penalties are much larger than earliness costs a due date sequencing rule is best. Another interesting observation that can be made from the data is that in spite of the dependence on FCFS sequencing in much of the literature, FCFS provided among the worst actual performance of the sequencing rules tested – it does however provide the best predictions of performance.

Rajasekera, Murr, et al. [21] open by observing that including more information into the dynamic flow time prediction process produces better results. Much of the paper subsequently focuses on an analytical description of a load-balancing algorithm that could be implemented in

an information system integrated with the manufacturing system. The authors conclude that after applying their load balancing procedure and assuming FCFS processing, then setting due-dates is straightforward even when taking into account the jobs already in the system. As a parting note, the authors concede that more complex work centers would require more complex queuing decomposition methods and further analysis.

Predictive use of DES Modeling

Much of the existing literature talks about using models of systems to conduct experiments where the objective is to optimize system performance by adjusting resources or queuing behavior [22, 23].

There is some literature that seeks to use the model to evaluate differing courses of action such as selecting a sequence of jobs to be scheduled. For example, Azzaro-Pantel, Bernal-Haro et al. describe using a combination of discrete event simulation and a genetic algorithm to optimally dispatch tasks in a job shop environment, with the genetic algorithm generating the sequences and the DES model evaluating each sequence [24]. In a related fashion, Reijers discusses using short-term simulations coupled with work flow to provide decision support, i.e. scheduling additional resources during peak loads [25]. Much less of the literature discusses the potential for use of the faithful model to make predictions about the system *just the way it is*. Rojanapibul and Pichitlamken make some excellent observations about using embedded simulations to calculate prediction intervals in a flow shop environment

[26]. Cates and Mollaghasemi describe the use of simulation to predict project completion dates and thereby enhance visibility of risk to better manage completion of complex projects [27]. In both of these cases, though, the job parameters were reasonably established before predictions were made.

Developmental Details

The author's prototype solution for implementing this methodology is composed of two distinct, but closely interrelated components. The first component, which replicates the previously mentioned manual analysis as an automated process, uses historical data to determine descriptive parameters. The second component is an embedded simulation model that makes use of these descriptive parameters to replicate the behavior of the target system. It is important to note that the predictive power of this construct is dependent on both components, which must act in concert.

Automated Analysis

The automated analysis component performs five major functions: (1) decompose the departure transactions (by job and by station) from the workflow system into Departure and Arrival events, (2) using the correlated Departure and Arrival events determine the rework rate of the sample of jobs by station, (3) using the correlated events by station, determine the queuing behavior for that station, (4) using the correlated events by station, decompose the

total time at a station for a job into waiting time and processing time and fit the processing times to a valid statistical distribution, and (5) utilizing the transaction logs, determine the inter-arrival rate per month. The last four functions output their results as a series of parameters to be used by the embedded simulation.

The first function is a pre-processing step facilitating the remaining functions. As mentioned, the system in question is an electronic workflow system. As such, there is no perceptible transportation delay. Without transportation delay, the decomposition of the departure transactions simply requires the creation of a departure event from the current station, and an arrival event at the next station visited by the job. The times of occurrence for each of these events are identical; the only complicated aspect is determining the next station visited. As this complication is purely self-inflicted by the author's implementation of transactions, recording the details of overcoming this particular hurdle will be glossed over. A sage practitioner would be well served to capture both the source and destination stations within the departure transaction and thus avoid this step entirely. As the output of this step is only used as the input for the subsequent three steps, there is no need to store these results back to the database.

The second function uses the correlated departure and arrival events created by the first function to determine rework rates. This is accomplished simply by implementing a two-level, nested, case construct which takes at the outer-level the source station, and at the inner-level the destination station. The rework status per job is then captured as a logical action, in

the author's case a job is accepted, rejected or returned without further action. The relative frequencies of these actions are recorded by station as model parameters in the database and are used by the branch components to correctly route jobs from one station to the next – this pairing of analytical and simulation components directly addresses $\vec{R}$ from Equation 5.2.

The third function, determining the queuing behavior, is considerably more interesting to describe, and is in fact, half of the novel aspect of the author's formulation for attacking $\vec{Q}$ in Equation 5.2. In general terms, the concept of the function is similar to executing a DES in reverse. In a normal DES, both the processing time for a job, and the queuing policy for a station are specified and the result for the job is the departure time from the station. In this case, however, the arrival and departure times are known and the results of the analysis are the processing time for the job, and the queuing behavior of the station. More specifically, the historical jobs arriving at a given station are processed in time-order of their arrival at the station but the jobs are placed in the queue based on their, known a priori, departure time. Executing this process one input job at a time, it is possible to determine the queue insertion location at the station, and the accumulated processing time for the job.

As an example of this process consider the following sequence: job 1, which arrives at station X at time 0 and is known to have departed at time 20, finds station X empty and idle; since the server is empty and idle, job 1 is immediately placed in service (location = 0, queue depth = 0) and begins to accumulate processing time. Job 2 (arrives at time = 5, will depart at time = 21) arrives at station X; since the station is not idle the departure time of the newly

arrived job is compared to that of the job in service; since job 2 will depart after job 1, it is placed in queue; since the queue is empty, job 2 is queued at location = 1, queue depth =1. Job 3 (arrives at time = 10, will depart at time = 30) arrives at station X; since job1 is still in service, departure times for jobs 1 and 3 are compared; job 3 will depart after job 1, so job 3 is queued; since job 3 will depart after job 2, it is queued after job 2 at location 2 and queue depth = 2. Job 4 (arrives at time = 15, will depart at time = 25) arrives at station X; since its departure time is after job 1 (still in service), job 4 will be queued; since job 4 will depart after job 2 and before job 3, it is queued at location = 2, queue depth = 3 which is recorded in Figure 24 as '2/3'. Executing this scenario, and stopping at time = 15 is represented graphically in Figure 24.

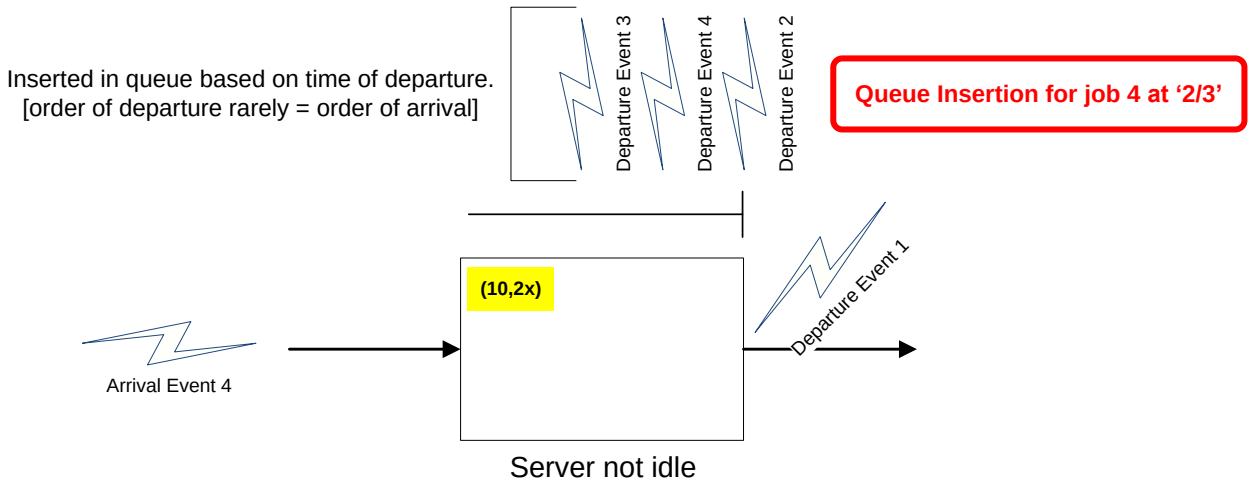


Figure 24 - Queue position determination

In pseudo-code, the virtual Server performs the following top-level tasks:

```
  Read previous 180 days of Transactions for Server;
  Create Arrival Events and Departure Events based on
transactions for completed jobs;
  loop through events in time order {
     if (arrival event) Push(event);
     else if (departure event) Pop(event);
  }
```

The virtual Server Push method performs the following:

```
  new_job = get_job_from_event(event);
  if (server idle)
   in_progress_job = new_job;
   new_job.arrival_location = 0;
   new_job.arrival_queue_depth = 0;
  else (server busy)
   if (new_job.departure < in_progress_job.departure)
      in_progress_job.add_processing_time_to_date();
      queue.add(in_progress_job);
      in_progress_job = new_job;
      new_job.arrival_location = 0;
      new_job.arrival_queue_depth = queue.size();
   else
      queue.add(new_job);
      new_job.arrival_location = queue.find(new_job);
      new_job.arrival_queue_depth = queue.size();
```

The corresponding virtual Server Pop method performs the following:

```
  in_progress_job.add_processing_time_to_date();
  if (queue not empty)
   in_progress_job = queue.next();
  else
   server idle = true;
```

The output of this function, which is accomplished by the “Push” method of the virtual server, is three parameters per station specifying the fraction of jobs that preempt, queue at

the head-of-line, and queue at the tail-of-line. Jobs that don't meet any of the three criteria are assumed to be randomly placed in the queue between head-of-line and tail-of-line.

The fourth function separates the processing time from the waiting time and then fits the processing times to a statistical distribution. This statistical distribution addresses, in conjunction with the server simulation component, the $\vec{P}$ component from Equation 5.2. In the author's implementation, the first portion of this function – separating processing and waiting times for a job at a station – is accomplished by a combination of the "Push" and "Pop" virtual server methods described above.

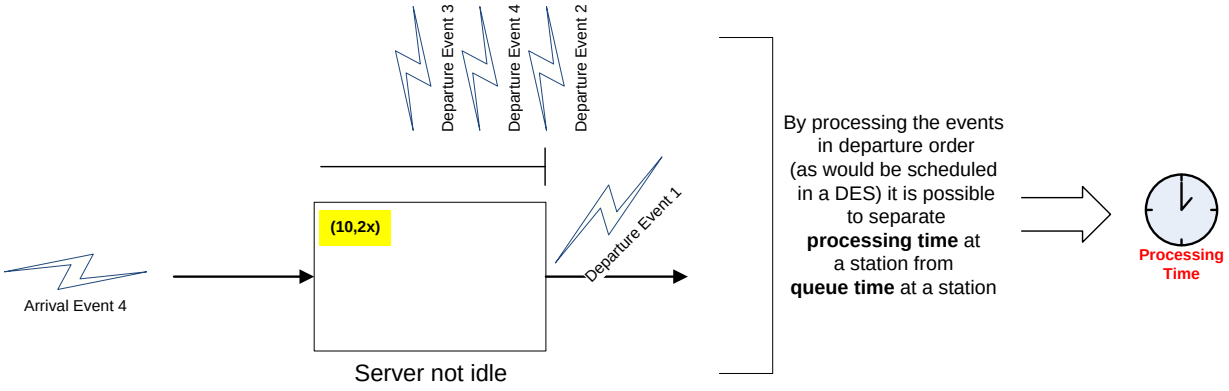


Figure 25 - Processing time determination

The second portion of the function uses a well known formulation to convolve the resulting processing times at a given station such that a linear, least-squares regression of the convolved data exhibits the shape and scale parameters of a Weibull distribution fitted to the unprocessed data. Similar to the implementation(s) above, the newly calculated parameters are combined using exponential smoothing – as in the second and third functions – with the existing parameter values and the resultant, smoothed values stored back into the database,

two parameters per station. Unlike the previous implementations above, however, a Kolmogorov-Smirnov goodness of fit test is executed between the source data and the fitted distribution, and the newly calculated parameters are only combined with the existing parameters if the test statistic is less than the adjusted critical value for the sample size [22].

As the reader may have already surmised, the fifth function, calculating the inter-arrival rates by month, when coupled with the source component of the simulation, completes the input parameters to Equation 5.2, namely IAT. This function is executed very simply using an SQL query which aggregates the arrivals by month for the previous 12 months. The more interesting aspects of this function reside in the simulation component discussed below.

Embedded Simulation

The Source component uses parameters from the database to implement a non-stationary, Poisson arrival process which varies month-by-month. At each arrival event the Factory Component (see below) is used to generate an order entity which is sent to the output component of the source which would normally be either a Branch or a Server.

The following pseudo-code initializes the non-stationary arrival process:

```
Query database for monthlyIAT[month];
hours = (lastDay[current_month] - today) * 24;
for (month = 0..6) {
   'IATBin<month>Hours' =  hours;
   'IATBin<month>Rate' = monthlyIAT[month];
   Hours += lastDay[month] * 24;
}
```

While the following pseudo-code implements the non-stationary arrival process:

```
in scheduleArrival()...
for (month in 0..6) {
   if (simulation time < IATBin<month>Hours) {
      IATRate = IATBin<month>Rate;
   }
}
IATgenerator.setRate(IATRate);
nextArrivaltime = simulation time +
                                    IATgenerator.draw();
```

The Factory component produces, on demand, entities of type Order with processing times per step drawn from Weibull distributions whose parameters are taken from the analytical component. The Factory is also capable of creating a special "target" Order.

The Order component extends the Entity class and implements the Comparable interface. It also contains a Properties object that is used to capture the history of the event as it traverses the model.

The Server component, in conjunction with its Queue, implements the empirical queuing behavior specified by the parameters from the analytical component.

Pseudo-code for Preemptive, LIFO, FIFO, and Random queuing:

```
new_job = get_job_from_event(event);
new_job.queue_behavior = uniform.draw();
if (server idle)
   in_progress_job = new_job;
   schedule_departure(new_job, new_job.process_time); else
 if (new_job.queue_behavior < preepmt)
    calendar.remove_depart_event(in_progress_job);
    in_progress_job.process_time -=
                        processing_time_to_date();
    queue.add(in_progress_job,HEAD_OF_LINE);
    in_progress_job = new_job;
    schedule_departure(new_job,
                       new_job.process_time);
 else if (new_job.queue_behavior < LIFO)
    queue.add(new_job,HEAD_OF_LINE);
 else if (new_job.queue_behavior < FIFO)
    queue.add(new_job,TAIL_OF_LINE);
 else
    queue.add(new_job,RANDOM_LOCATION);
```

The Queue component utilizes the CompareTo() method of the Order entities to queue the Orders based on the value set for the Order by the Queuing Behavior method of the server.

The Branch component implements routing of incoming Orders to one of two or more destinations based on the rework parameters form the analytical component. The author's implementation adds special treatment for the "target" Order – it is not allowed to exit through the "return without further action" sink.

The Sink component disposes of non-target Orders as they depart the simulation, and store the target Orders in a static collection when they exit. The Sink also signals a SimulationEnd event when the target Order exits.

System Under Test

The model of the system under test is implemented as a top-level simulation object. This object has one source component implementing non-stationary arrivals as indicated above and containing an order factory producing orders in accordance with the processing time distributions based on the Weibull $\vec{P}$ parameters, including the special "target" order. The simulation object instantiates 11 servers which, in conjunction with their attendant queues, implement empirical queuing behavior in accordance with the $\vec{Q}$ parameters from the analytical component. It also instantiates 11 branches (3-way) that implement rework based upon the $\vec{R}$ parameters. Finally, the simulation implements two sink components, one for capturing objects successfully traversing the system and a second for objects that are returned to the customer without further action.

These components are instantiated, logically connected as pictured in Figure 23, initialized with the parameters as mentioned above, the queues pre-loaded with jobs according to the current date's queues. At this point the target job is introduced to the system, and the simulation clock started. The simulation run terminates when the target job exits via the first sink.

To facilitate statistical analysis, the target jobs from each replication of the simulation are maintained until the desired number of replications has been executed. At that point the

collection of target jobs can be summarized, in this case by determining the upper confidence limit for the mean of the turn-around time.

Test Methodology

As the actual system under test is, in fact, a transactional workflow system, it is possible to roll the systems state back to any point in time covered by the transaction log. Utilizing this capability it is possible to (1) determine actual turn-around times for jobs entering the system on any given day, and (2) to execute both the analytical and simulation components against the data that was available on that same day. With both data sets available simultaneously it is possible to compare the actual and predicted data side-by-side.

The actual turn-around times were gleaned from the workflow system through an SQL query of the database that provides persistence to the workflow system. This query was structured such that the output consisted of the date, the mean turn-around time of the jobs that entered the system on that date, and the number of jobs entering on that date. Using this data it was then a simple bit of manipulation in Microsoft Excel to generate a time-weighted average turn-around time looking back 10 days to smooth the necessarily jagged plot of mean turn-around times.

The predicted turn-around times were generated by providing a “main” function that specified a date for simulation such that the analytical component could execute as if it were that date and looking 180 days into the past to calculate the simulation parameters, and then

using that same date, the simulation component could execute 200 replications of the model capturing the upper confidence limit (UCL) of the mean turn-around time. After the analytical and simulation components had executed for the date specified, the date was incremented by 1 and the process repeated until the desired end date was reached. The output of the components was adjusted such that the output was the date, the number of jobs in queue on that date, and the UCL of the time in system for a new job on that date.

With the two data sets described it is a simple matter to match the actual data and the predicted data by date, again using Microsoft Excel.

Results

Initial results of the tests conducted indicate an expected result – that the turn-around time predicted for a given job is closely correlated ($\rho = 0.76$) to the number of jobs in queue when the new job enters the system as shown in Figure 26. The red line in the figure represents the 90% UCL for the mean turn-around time predicted by the model, while the blue line – plotted against the secondary y-axis – represents the total number of jobs in the system when the target job arrives. The correlation is not perfect due to the location of the jobs in the system. If, for example, the 100 jobs in the system are evenly distributed across the 11 servers, then one would reasonably expect that the target job would end up getting queued in several of the steps along its processing journey. The results would be very different if most of the 100 jobs were about to exit the system, perhaps at server 11. In this case the target job would race

through stations 1 through 10 without queuing (unless previous jobs were inserted, due to rework, into previous queues), not slowing until step 11. And, depending on the relative processing times for the 100 jobs queued at step 10, it is possible, though unlikely, that the target job could run through the entire system without experiencing any queuing whatsoever.

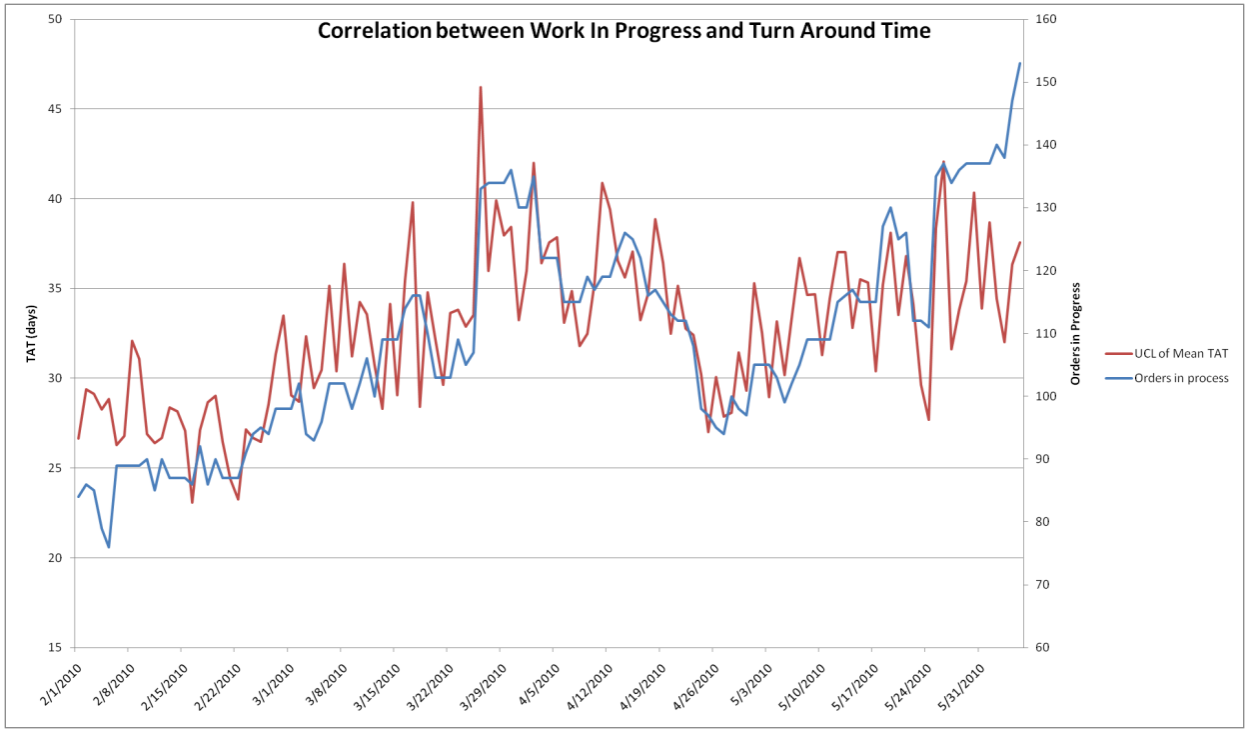


Figure 26 - Correlation between WIP and TAT

Of more practical benefit is the indication of a good correlation between the predicted turn-around times for a given day, and the actual, observed turn-around times for jobs entered on that day as shown in Figure 27. The red line is the same as in Figure 26 – the 90% UCL for

the mean, but the green line represents the mean turn-around time for the actual jobs that entered the system on that day.

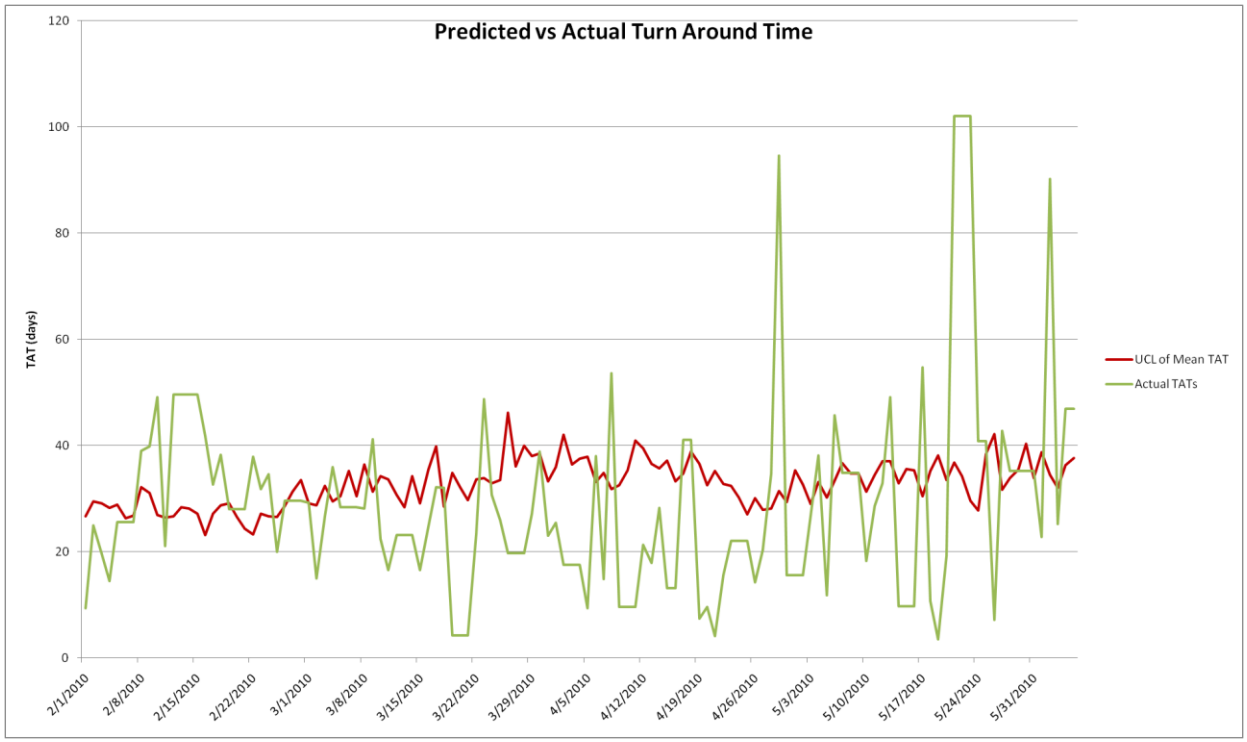


Figure 27 - Predicted versus Actual TAT

The performance indicated in Figure 27 above is actually quite good. Simply using the UCL of the mean flow time for predicting the due dates yields a service level of approximately 65%. Adjusting the flow time by adding in some multiple of the standard deviation of the forecasting error e_i ($1.285\sigma_e$) allows the achievement of a 90% service level. And while achieving at least a 90% score is desirable for the process owner, it may be more attractive to a customer to tune the predictive subsystem for an 80% service target ($0.841\sigma_e$) and incentivize

the process owner to achieve the next 10%. An interesting side benefit of this methodology is that it provides a ready mechanism for continuous improvement, i.e. if the processor is successful in achieving 90% during this period, future job flow times will be based on this tighter standard.

Conclusions

The author's previous work indicated that the existing, deterministic methods of quoting due dates suffered when applied to systems not based on FCFS queuing and argued that investigation of a stochastic approach was warranted. This paper documents that investigation, and indicates that a carefully crafted mix of automated analytics and embedded simulation might indeed provide a practical alternative for higher-fidelity due date quoting in systems with non-standard queuing behavior and high levels of rework. The author is currently performing additional research based on a prototypical implementation integrated to a production workflow system to validate these results in a practical setting.

In the experiment described in the following chapter, this research was extended to incorporate the error distribution described in the results section, above.

CHAPTER SIX: REAL-TIME ASSIGNMENT OF DUE DATES WITHIN WORKFLOW MANAGEMENT SYSTEMS

The following material has been accepted for presentation and publication at the 2011 Institute of Industrial Engineers (IIE) Industrial Engineering Research Conference (IERC) [28].

Abstract

This research presents the application of real-time simulation to assign due dates within a multiprocessor, electronic workflow management system. The workflow system under study accepts from customers external requests (called orders) for work to be done. Upon receiving an order from a customer, the workflow system immediately quotes that customer a date by when the review of the order will be completed and a customized proposal against the order is generated. The customer fully expects the review of the order to be completed by the due date, and severe penalties are incurred if the review is completed before or after the quoted due date. Therefore, accurate determination of due dates for the delivery of this service is critical. The authors present an innovative approach to perform real-time sequencing of customer orders. Using machine learning concepts and discrete event simulation, the approach minimizes the deviation between actual proposal delivery dates and the quoted due dates.

Introduction

Today, organizations face unprecedented levels of intense competition and these organizations are motivated to improve their competitive advantage through increased productivity, improved customer service and strict conformity to standards. As a result, information technology solutions that support and automate internal business processes have become critically important and serve as the backbone of the modern-day firm. These business processes, which describe key procedures within an organization, often involve multiple steps, several people, and significant resources. Workflow is the term that describes the logical steps that comprise a business process, i.e., the sequence of steps and the required tasks, resources (people and machines), tools and information needed for each step. It is this sequence of steps that creates or adds value to a firm's activities.

The information technology software solutions that support the automated coordination of the steps of a business process are called workflow management systems (WFMSs). The modern WFMS is a computerized system that is composed of a set of applications and tools that helps to define, create, and manage the tasks, resources, tools, and information associated with the workflows. WFMSs are generally responsible for the scheduling and execution of the tasks associated with the processes, where the core capabilities supported in most of today's workflow technology solutions are: database management, document management, project management, electronic messaging, and directory services. For example, in a manufacturing environment, a product design specification originating from design engineer might be

automatically routed for approval through the WFMS to the project leader then to a technical director then to the production engineer and then back to the initiating design engineer. At each step in the design specification document workflow, one individual or a group of people is responsible for a specific task.

At each step within its workflow, the order can be placed in one of four positions in the queue of orders: (1) at the head (first) position of the queue, (2) at the tail (last) position of the queue, (3) at a random position in the queue, or (4) it can preempt the order that is in process at the step. Once the task is complete, the workflow management system ensures that the individuals responsible for the next task are notified and receive the information they need to execute their associated steps of the process. It is important to note that, if a correction to an order needs to be made, it is sent to previous steps to be reworked, before it continues through its workflow. The nature of a WFMS depends on the type of workflow that is to be supported – either content-based or activity-based. Content-based workflow places a content object (e.g., a document) as the focal point of the process. Activity-based workflow focuses on a task. The focus of this research is content-based workflow.

Description of the Problem

The WFMS that inspires this research is a content-based, multiprocessor, electronic production workflow management system. The system accepts external customized requests

(called orders) from customers over time for work to be done. Upon receiving an order from a customer, the workflow system immediately quotes that customer a date by when the review of the order will be completed and a customized proposal against the order is generated. The customer fully expects the review of the order to be completed by the due date, and severe penalties are incurred if the review is completed before or after the quoted due date. The customers demand both demonstrably strict dates – that is to say that orders should not be delivered *significantly* before quoted due dates as this lends the impression that the due dates have been over-inflated, detracting from the credibility of this methodology. Moreover, penalties for not meeting quoted delivery dates tend to be severe as they effect the likelihood of customers accepting the final order. Therefore, accurate determination of due dates for the delivery of this service is critical, and the desire is to minimize the deviation between actual proposal delivery dates and the quoted due dates, or the mean squared lateness. However, accurate determination of due dates for the delivery of customized work based on non-technical specifications is a challenging task, and due date assignment is simply a difficult problem given the dynamic nature of most productive environments.

In this chapter, the authors propose a new due date assignment method, where the method uses real-time simulation to predict the actual delivery date of the customized work to the customer. As can be imagined, the queue priority discipline at each step, i.e., the position in which the order is placed in queue at each of its steps as the order progresses through its

workflow, greatly influences the order's delivery date. Therefore, it is imperative that any due date quoting approach consider this in its prediction.

The remainder of this chapter is organized as follows. Section 2 summarizes the previous research highlighting some of the salient literature from the areas of due date quoting, and predictive use of simulation models. Section 3 presents the formulation of the problem under study. Section 4 describes the proposed due date assignment methodology, and Section 5 illustrates the performance of the proposed method within a real-world WFMS. The chapter is concluded in Section 6 with a summary and a discussion of future research.

Previous Related Work

Cheng and Gupta [14] survey the existing research with respect to due date determination. In this survey, Cheng and Gupta [14] open by pointing out that meeting due dates is extremely important to practicing managers due to the customer service implications. They then utilize a classification scheme first proposed by Elion [15], which has six dimensions: (1) Static vs. Dynamic, (2) Deterministic vs. Stochastic, (3) Single-product vs. Multi-product, (4) Single-processor vs. Multi-processor, (5) Theoretical vs. Practical, and (6) Exogenous due dates vs. Endogenous due dates. Since exogenous due dates obviate due date quoting and lead directly to sequencing and scheduling problems, Cheng and Gupta [14] focus their attention on endogenous due dates. Using the above classification scheme, they conclude that there is very little extant research on dynamic, complex, multi-processor systems.

Subsequent to the survey conducted by Cheng and Gupta [14], Cheng [17] describes a sequencing algorithm when using the slack due date quoting policy. He simplifies the system under consideration by assuming that once a set of jobs is sequenced, no subsequent jobs will affect the system's performance, there will be no re-sequencing of the jobs between stations and all of the earliness and tardiness costs are constant. In effect, the lack of consideration of dynamic arrival of jobs and non-permutation scheduling becomes a presupposition of first come, first serve (FCFS). Cheng [17] concludes that a shortest processing time (SPT) sequence is optimal, although this conclusion does not fully support the findings of Duenyas and Hopp [18], who propose an analytical framework for evaluation of various job sequencing rules given that flow times can be optimally predicted. Working through a series of increasingly generalized scenarios, they conclude that an earliest due date (EDD) sequence is optimal if the tardiness penalty is constant for all customers and proportional to the tardiness, which seems to contradict Cheng [17]. To achieve this result Duenyas and Hopp [18], only assume that preemption does not take place.

Similar to Duenyas and Hopp [18], Lawrence [19] presupposes that the practitioner either has a simple system with closed-form flow time estimates, or has a method to determine flow time for complex systems. With that as a precondition, he describes an analytical approach to setting due dates based on previously-observed forecasting errors. While Lawrence [19] proposes to fit the forecasting errors, which he refers to as "*G*", using a Ramberg-Schmeiser distribution, he concludes that Erlang and Gaussian distributions worked equally well. He makes

a key observation that is particularly germane in this context. Various measures of performance lead to differing uses of the error distribution. For example, mean absolute lateness is minimized by adding the median of the error distribution to the predicted flow time. Mean squared lateness is minimized by adding the mean of the distribution to the predicted flow time, and service level matching is met by adding the target percentile of the distribution to the predicted flow time, e.g., $G^{-1}(0.9)$ for a 90% service level.

Van Ooijen and Bertrand [20] introduce a distinction in terminology intended to allow some leeway between the tightly-estimated Internal Due Date (IDD) and the slightly looser External Due Date (XDD). The difference between the two is analogous to a margin of error e_i, Alfieri's Safety Time, or Lawrence's G. The authors propose to adjust the XDD using the ratio of the current level of work in progress (acwip) to the average level of work in progress (nwip). The results of Van Ooijen and Bertrand [20] bring some closure to the disagreement between Cheng [17] and Duenyas and Hopp [18] by noting that when earliness and lateness penalties are of similar magnitude, then SPT sequencing works best; however, when tardiness penalties are much larger than earliness costs, a due date sequencing rule is best. Another interesting observation that can be made from the data is that, in spite of the dependence on FCFS sequencing in much of the literature, FCFS is among the worst performers of the sequencing rules tested. It does, however, provide the best predictions of performance.

Much of the existing literature discusses using models of systems to conduct experiments, where the objective is to improve system performance by adjusting resources or queuing

behavior [22, 23]. There is some literature that seeks to use the model to evaluate differing courses of action such as selecting a sequence of jobs to be scheduled. For example, Azzaro-Pantel, Bernal-Haro et al. [24] describe using a combination of discrete-event simulation and a genetic algorithm to optimally dispatch tasks in a job shop environment, with the genetic algorithm generating the sequences and the DES model evaluating each sequence. In a related fashion, Reijers [25] discusses using short-term simulations coupled with workflow to provide decision support, i.e., scheduling additional resources during peak loads. Much less of the literature discusses the potential for use of the faithful model to make predictions about the system just the way it is. Rojanapibul and Pichitlamken [26] make some excellent observations about using embedded simulations to calculate prediction intervals in a flow shop environment. Cates and Mollaghasemi [27] describe the use of simulation to predict project completion dates and thereby enhance visibility of risk to better manage completion of complex projects. In both of these cases, though, the job parameters are reasonably established before the predictions are made.

This review of the literature illustrates the bounds of the current literature and highlights the lack of coverage for due date quoting in systems (in research and in practice) that do not implement rigid queuing disciplines and where job preemption and job recirculation is permissible and commonplace.

Problem Formulation

We now provide the formulation of the due date quoting problem as it relates to the WFMS in this study. First, however, the relevant notation is given.

Notation:

N: set of orders to be scheduled and for which due dates are quoted, where order $i = 1, \ldots, |\mathbf{N}|$

$\mathbf{S}_i$: set of steps for customer order i, where step $j = 1, \ldots, |\mathbf{S}_i|$

$\mathbf{M}_j$: set of processors at step j

r_i: release date for customer order i, i.e., the date that order i arrives to the system to receive a due date quote

p_{ij}: processing time for order i at step j in its workflow

w_{ij}: waiting time for order i at step j

$\hat{d}_i$: quoted due date for order i

e_i: margin of error associated with order i

The estimated, or quoted, due date $\hat{d}_i$ of an order i is a function of four key elements, as shown in Equation 6.1,

$$\hat{d}_i = r_i + \sum_{j=1}^{|\mathbf{S}_i|} p_{ij} + \sum_{j=1}^{|\mathbf{S}_i|} w_{ij} + e_i. \tag{6.1}$$

Each term in Equation 6.1 is either obtained from source data or derived from the uncertainty of the data. The quoted due date for an order *i* is a function of its release date r_i. The quoted due date for order *i* is also a function of its processing times at its $|\mathbf{S}_i|$ workflow steps, $\sum_{j=1}^{|\boldsymbol{S}_i|} p_{ij}$, where the actual processing time p_{ij} values are drawn from a random probability distribution. The error e_i may be assumed or estimated from historical performance, and the waiting times of order *i* $\sum_{j=1}^{|\boldsymbol{S}_i|} w_{ij}$ are a function of the number of orders in queue at each step and the queuing discipline at each step. The salient difficulty in predicting completion times, i.e., turn-around times (TATs), which ultimately determine due dates, in a system with stochastic processing times and dynamic queuing priority disciplines is summarized in Equation 6.2,

$$W_i \equiv \sum_{j=1}^{n_i} w_{ij=} f\left(IAT, \overrightarrow{P}, \vec{Q}, \vec{R}\right), \tag{6.2}$$

where $f\left(IAT, \overrightarrow{P}, \vec{Q}, \vec{R}\right)$ is the waiting time function for order *i*, and the order waiting time is a function of IAT, which is the interarrival time for orders that arrive to the system after order *i*, and $\overrightarrow{P}, \vec{Q}, \text{and} \vec{R}$, which are the vectors of processing times, queuing priority disciplines, and rework probabilities, respectively, at each step for the other orders in the system. Note that the order arrival process need not be stationary. Estimating W_i is the greatest challenge in quoting due dates due to the inherent stochastic nature of the WFMS.

The authors' proposed method to determine W_i involves: (1) constructing an embedded discrete-event simulation (DES) model, (2) determining the parameters for the DES model that

are applicable when a new order *i* enters the WFMS, (3) determining the properties of order *i* necessary for representation within the DES model, and (4) repeatedly running the model until an acceptable margin of error on predicting its TAT, thus, its estimated delivery date, is determined. The measure of performance is the mean squared lateness, or

$$\overline{L^2} = \frac{1}{|N|}\sum_{i \in N} L_i^2 = \frac{1}{|N|}\sum_{i \in N}\left(c_i - \widehat{d}_\iota\right)^2,$$

where c_i is the actual completion time of order *i*, and $c_i = r_i + \text{TAT}_i$.

Proposed Methodology

The authors now describe the proposed due date assignment methodology and its three main phases – (1) Update, (2) Record, and (3) Simulate. However, before describing the methodology, the assumptions on which it is based are provided. The proposed methodology is developed based on the following list of assumptions: (1) there is exactly one processor at each step, (2) there is no forced idle time at the processors at the steps, and (3) the processor times at each step follow a Weibull distribution.

Phase 1 of the proposed method, the Updating phase, uses historical order data from the WFMS. The number of past orders *n* or the past *t* time periods is used to update the parameters of the embedded DES model by executing the heuristic developed that effectively reverses the discrete-event simulation and records the behavior of the WFMS using the

historical data. The desired value of *n* or *t* is set by the user of the WFMS. The parameters of the DES model that are updated include the Weibull shape α and scale β parameters for the processing times, and the order rework probabilities at each step. Most importantly, the queuing discipline at each step is determined. Recall that, at each step within its workflow, an order can be placed in one of four positions in the queue of orders: (1) at the head (first) position of the queue, (2) at the tail (last) position of the queue, (3) at a random position in the queue, or (4) it can preempt the order that is in process at the step. A probability parameter P_k for each position *k* at a step is computed based on the historical order data, and $\sum_{k=1}^{4} P_{kj} = 1$ for each step *j*.

Each of the DES model parameter values is exponentially smoothed against the previously stored values using the smoothing parameter α. For the processing time parameters, however, an additional step is executed before the exponential smoothing. A Komolgorov-Smirnov Goodness of Fit test is performed for the newly-calculated distribution to ensure that the new parameters fit the processing time distribution. If they fit (with $\alpha = 0.05$), the exponential smoothing takes place. If the parameters do not fit, the new processing time values are discarded, and an exception is logged. After the DES model parameter values are updated, they are stored in a centralized database for later reference and updating. This updating phase occurs at a frequency *F* set by the user of the WFMS.

Phase 2 of the proposed method, the Record phase, records a "snapshot" of the current orders in the WFMS whenever a new customer order *i* arrives to the system. This snapshot

records the current orders that are in queue at each step as well as which processors are busy. These orders are used to populate the queues at the steps in the embedded DES model. Additionally, the past *n* orders (or the orders that arrived during the past *t* periods) are used in Phase 3 to inform the non-stationary arrival process.

Finally, Phase 3, the Simulate phase, places the arriving order *i* in the first queue in its workflow either at the head (first) position, at the tail (last) position, at a random position or the order preempts the order currently in process at the step based on the queuing probability parameters for that step, as determined in Phase 1. With the embedded model now loaded to match the current system's state, and the new order *i* inserted, the simulation model is run using the historical *n* orders (or the orders that arrived during the past *t* periods) until the new order *i* completes all of its $|\mathbf{S}_i|$ workflow steps.

A user-specified number of replications *R* are run, and the average completion times (and associated confidence intervals) for the new order *i* at each step are recorded. After the replications are completed, the step completion times are summarized, including the completion time of the last step in order *i*'s workflow. This value is the predicted value for the TAT, and ultimately the quoted due date $\hat{d}_i$, for the new order.

Experimental Study

Description of WFMS under Study

The proposed due date assignment methodology is evaluated within a workflow management system that supports a real-world business process and one that inspired this research. It is similar in logic to a reentrant flow shop in which the sequence of steps that an arriving order passes through is known and orders may return to previous steps (based on a probability) before exiting the system. There are 11 steps in the workflow that this particular WFMS supports (see Figure 28).

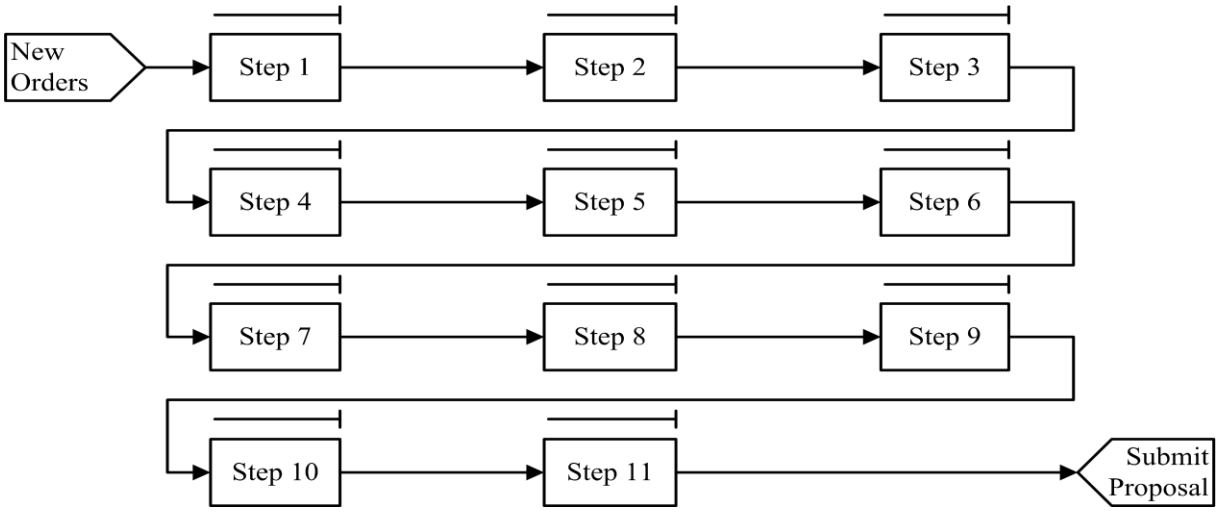


Figure 28 - Workflow sequence of orders in the WFMS under study

The Updating phase occurs once per day, i.e., the updating frequency *F* = 1 day. The historical data used to update the parameter values for the embedded DES model is from the past six months of data, i.e., *t* = 6 months, which uses approximately 145 days of production

workflow logs (from 2/1/2010 to 6/6/2010). During this period, 572 orders are received, processed, and returned to the originating customer.

Experimental Data

After each order arrives to the system, it is assigned a vector of processing times, which are derived from the historical order data. The processing time distributions for this experimental study are summarized in Table 3, and, in fact, the times can be described by the Weibull distributions fitted with $\alpha = 0.05$.

Table 3 - Processing times by step, which are derived from historical data

Step	Processing Time Distribution [WEIB(Scale β, Shape α)]
1	WEIB(0.15, 0.39)
2	WEIB(1.19, 0.44)
3	WEIB(0.40, 0.30)
4	WEIB(0.71, 0.41)
5	WEIB(0.93, 0.42)
6	WEIB(0.57, 0.34)
7	WEIB(0.82, 0.39)
8	WEIB(0.51, 0.34)
9	WEIB(0.34, 0.33)
10	WEIB(0.92, 0.41)
11	WEIB(1.32, 0.46)

The logic of the current WFMS presents to the processor at each step a list of the orders requiring processing with the newest orders at the top of the list. In other words, the WFMS processes an order at each step in last in, first out (LIFO) order. As each order is completed, the error between the predicted and actual flow times is captured and the standard deviation of the expanded sample is re-calculated. The upper confidence limit of the mean TAT is also calculated for each new order.

Discussion of Results

Initial results of the experiments conducted indicate an expected result – that the predicted TAT for a given order is closely correlated ($\rho = 0.76$) to the number of orders in queue when the new order enters the system as shown in Figure 29. The red line in the figure represents the 90% Upper Confidence Limit (UCL) for the mean TAT predicted by the model, while the blue line – plotted against the secondary *y*-axis – represents the total number of orders in the system when the new order arrives. Of more practical benefit is the indication of reasonable predictive performance (65% of the actual jobs were delivered before the predicted date) of the predicted TATs for a given day, and the actual, observed turn-around times for orders entered on that day as shown in Figure 30. The red line is the same as in Figure 29. The 90% UCL for the mean, but the green line represents the mean TAT for the actual orders that entered the system on that day.

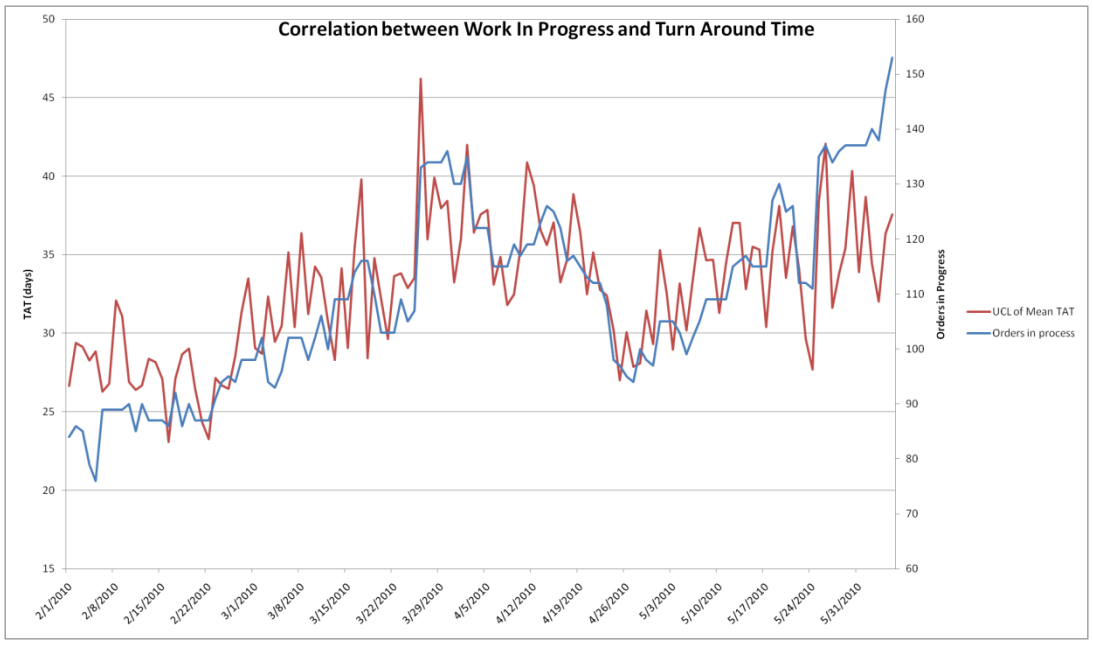


Figure 29 - Correlation between customer order WIP and TAT

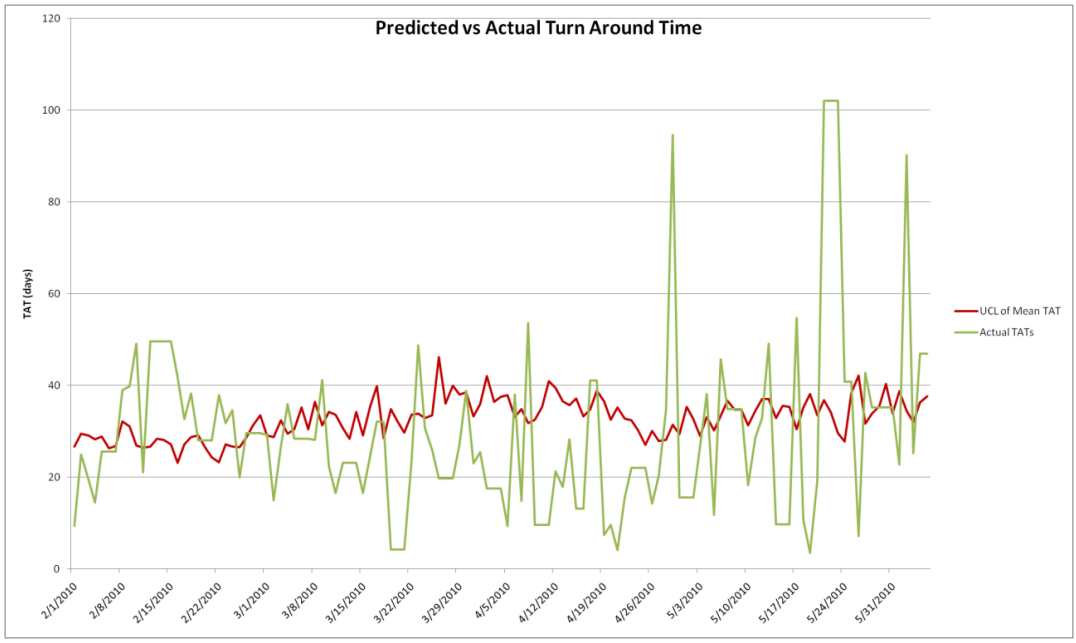


Figure 30 - Predicted TAT vs. Actual TAT

The unadjusted performance shown in Figure 30 is actually quite reasonable. Simply using the UCL of the mean flow time for predicting the due dates yields a service level of approximately 65%. Adjusting the flow time by adding in some multiple of the variance of the forecasting error e_i ($1.285\sigma_e$) allows the achievement of a 90% service level. Figure 31 depicts the same actual due date performance (green line) versus the error-adjusted predicted due date (in red). The implementation of Lawrence's methodology achieved 92% during the historical period analyzed. And, while achieving at least a 90% score is desirable for the process owner, it may be more attractive to a customer to tune the predictive subsystem for an 80% service target and incentivize the process owner to achieve the next 10%. An interesting benefit of this methodology is that it provides a ready mechanism for continuous improvement, i.e., if the processor is successful in achieving 90% during this period, future order flow times will be based on this tighter standard.

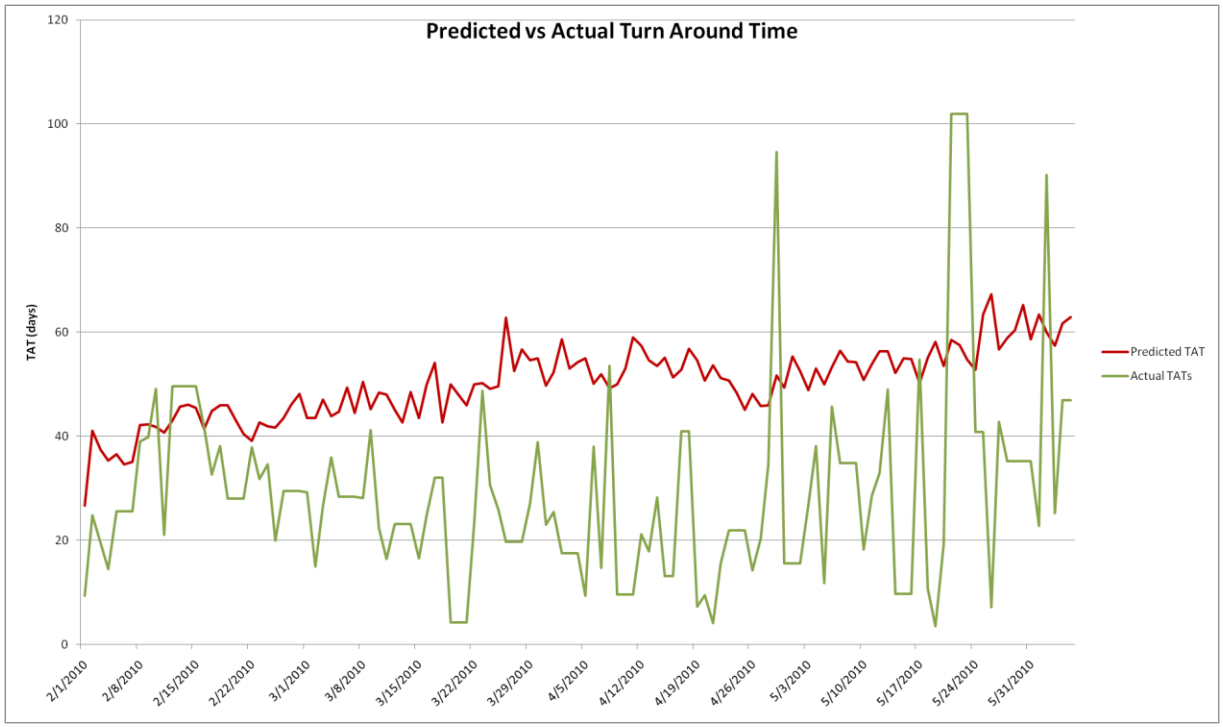


Figure 31 - Predicted versus Actual flow time

Summary and Future Work

The authors' previous work indicates that the existing, deterministic methods of quoting due dates suffer when applied to systems not based on FCFS queuing and argues that investigation of a stochastic approach is warranted. This paper documents that investigation, and indicates that a carefully-crafted mix of automated analytics and embedded simulation might indeed provide a practical alternative for higher fidelity due date quoting in systems with non-standard queuing behavior and high levels of rework. The authors are currently performing

additional research based on a prototypical implementation integrated to a production WFMS to validate these results in a practical setting.

Future work includes publication of a thorough description of the heuristic developed to decompose the WFMS historical logs, and analysis of the most appropriate exponential smoothing constant α, which the authors suppose will vary with the number of historical data points available and which are used to determine the DES modeling parameters.

The following chapter describes the extension of this research to encompass the live, production workflow system operating in real-time, with results reported after 75 days of operation during which 119 orders were processed.

CHAPTER SEVEN: RESULTS OF INTEGRATING MACHINE LEARNING AND SIMULATION TO PREDICT DELIVERY TIMES UNDER UNCERTAINTY

The following material has been submitted for review in the *Information Systems Frontiers* journal.

Abstract

This research presents a methodology for, and the results of a prototypical implementation of the application of real-time simulation to assign due dates within a multiprocessor, electronic workflow management system. The workflow system under study accepts orders from external customers for work to be done. Upon receiving an order from a customer, the workflow system's embedded simulation immediately quotes that customer a date by when a customized proposal against the order will be generated. The customer fully expects to receive the proposal by the due date, and severe penalties are incurred if the proposal is delivered after or significantly before the quoted due date. The customers demand both demonstrably strict dates – that is to say that orders should not be delivered *significantly* before quoted due dates as this lends the impression that the due dates have been over-inflated, detracting from the credibility of this methodology. Moreover, penalties for not meeting quoted delivery dates tend to be severe as they effect the likelihood of customers accepting the final order. Therefore, accurate determination of due dates for the delivery of this service is critical. Using machine learning concepts including a heuristic algorithm for

determining queuing behavior and discrete-event simulation including a component that implements non-standard queuing, the approach minimizes the deviation between actual proposal delivery dates and the quoted due dates.

Introduction

Accurate determination of due dates for the delivery of bespoke items based on non-technical specifications is a challenging task. Limiting fixed staffing levels to control costs is at odds with having sufficient resources necessary to reliably quote these due dates in a timely fashion. An environment that is extremely contentious with respect to the necessary resources and offering little in the way of firm prioritization only exacerbates the situation. And finally, when customers demand both demonstrably strict dates and penalties for exceeding those dates the situation becomes nearly untenable. The authors propose that an artful combination of automated analysis and efficient simulation might be successful in resolving this stark situation.

Prerequisites

In order to apply the methodology described here, a practitioner should already have (1) codified the business process to be modeled, (2) developed a functional, transaction-based workflow system, (3) performed an initial, manual data analysis of the processing times,

queuing behavior and rework rates, and (4) built a representative discrete event simulation (DES) model of the workflow process to validate understanding of the practitioner's system.

The diagram at Figure 32 represents a stylized representation of the business process under consideration showing the documents that map to the order and proposal and the actors involved in the process.

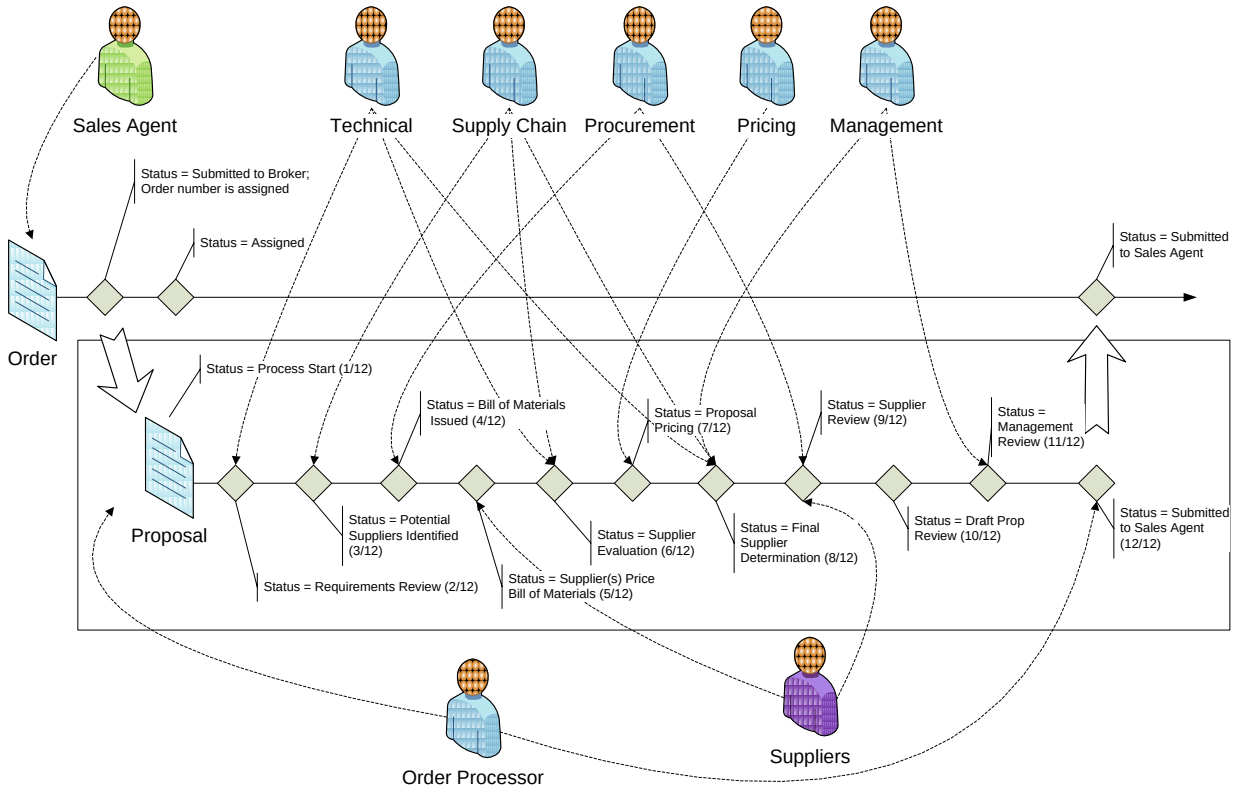


Figure 32 - Stylized business process

With the business process identified, it is then mapped to a workflow system that facilitates the flow of information, enforces the business logic, and functions as a common tool for situational awareness. This mapping is shown, conceptually, in Figure 33.

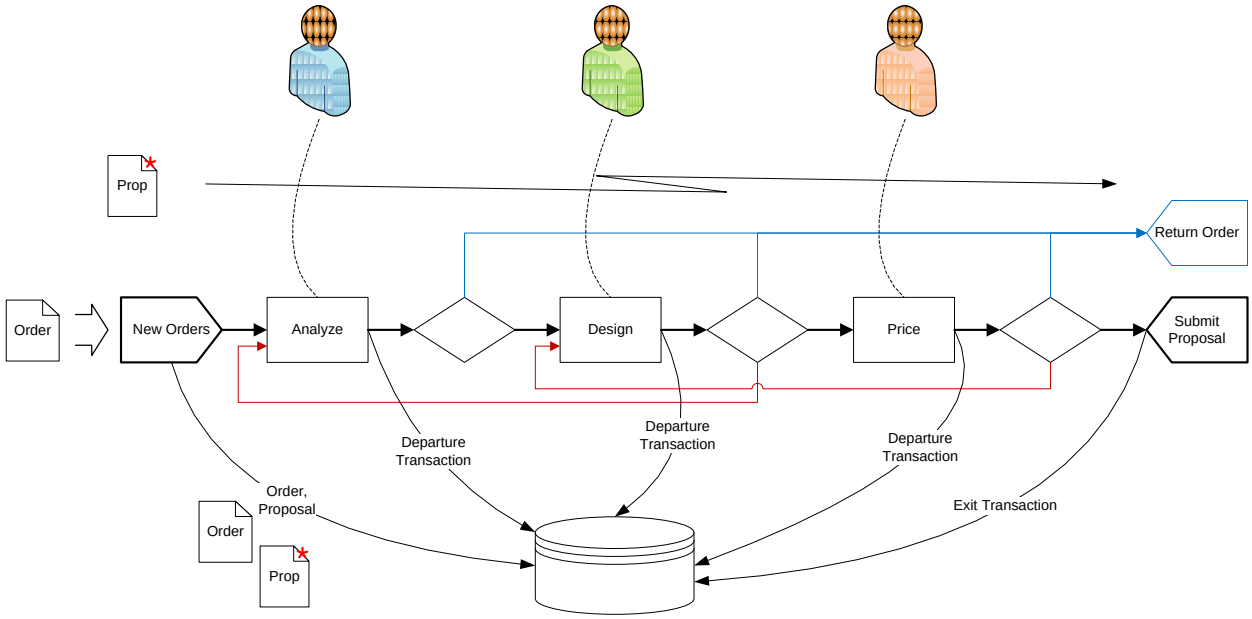


Figure 33 - Mapping the business process to the workflow system

In order to undertake the analysis of the workflow system's performance, the transactional events from the workflow system are decomposed into arrival and departure events. In the author's case, SQL queries and Java code were written to facilitate this decomposition which is depicted at Figure 34.

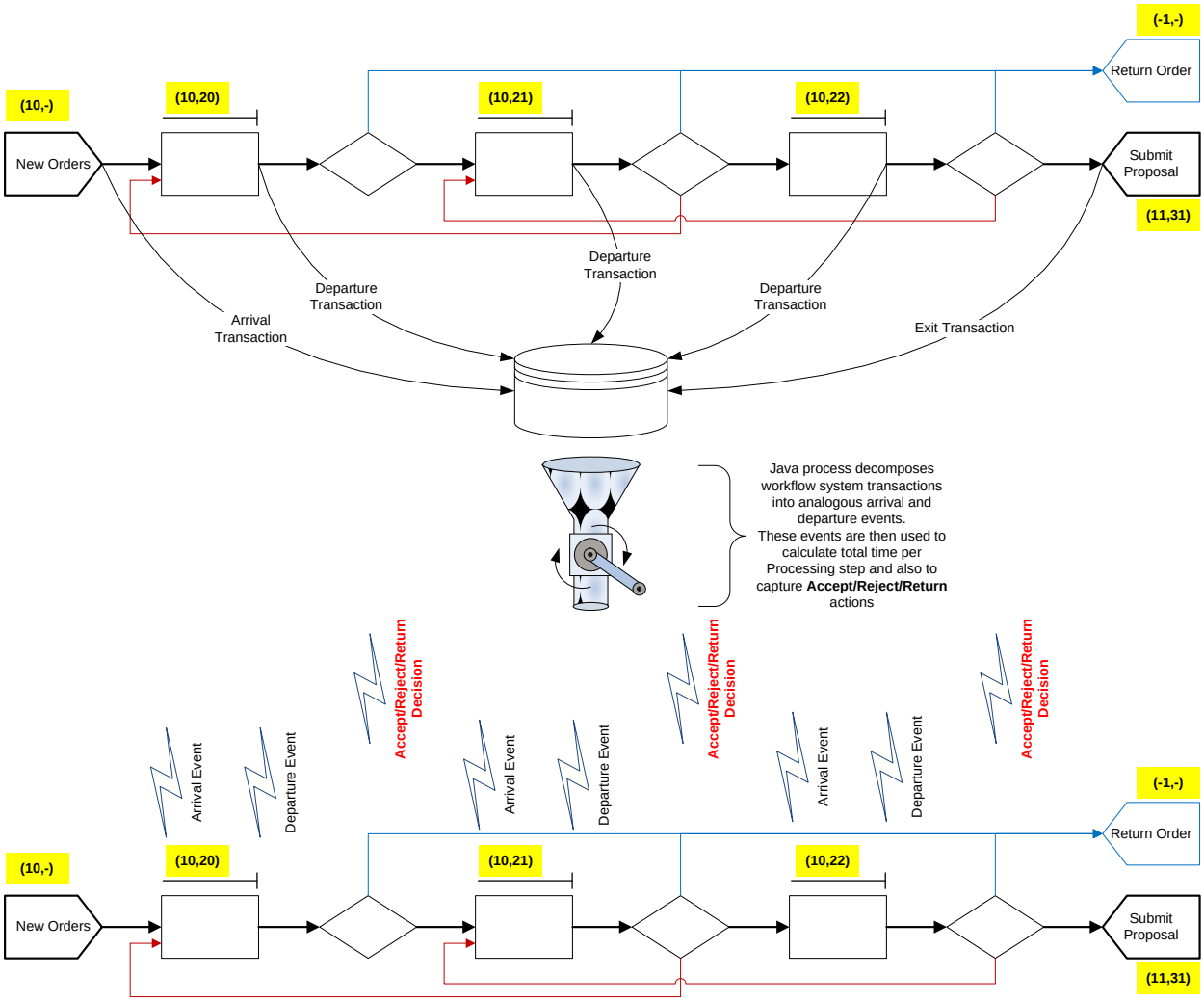


Figure 34 - Transactions to events

With the decomposed events as inputs, three distinct analytical steps are undertaken to determine the queuing behavior at each step, the processing times for orders at each step, and the re-work rates per step. The results of these analyses are combined and stored as parameters that will be inputs to both the stand alone and embedded DES models. This process is depicted at Figure 35.

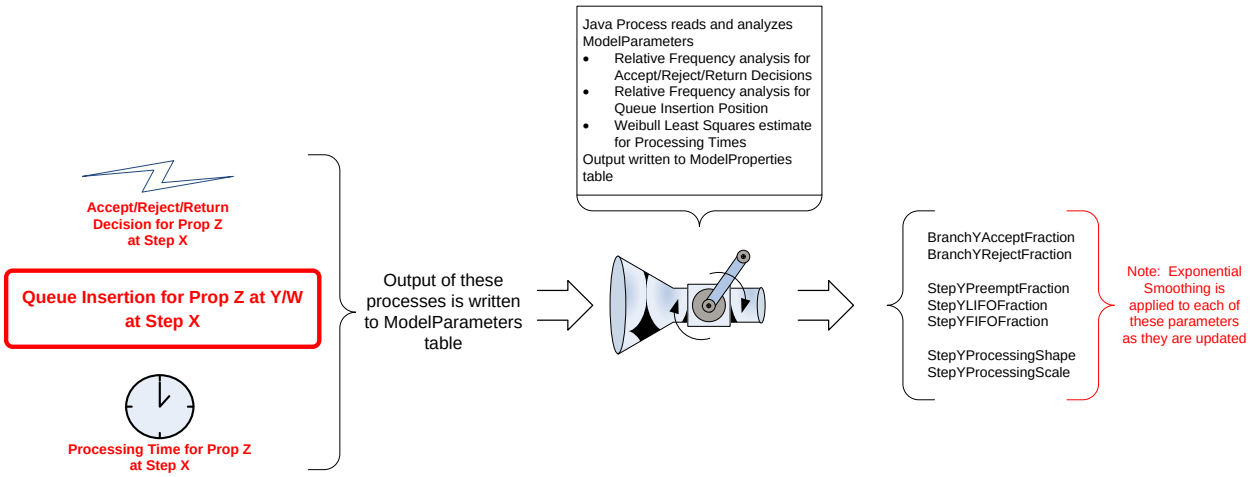


Figure 35 - Consolidation of analytical results

In this authors' case, the manual analysis was completed with some interesting results which will be detailed in the section titled "Necessity of Real-world Queuing Behavior" and which precluded a complete validation of the standalone model's behavior as in queue preemption is not readily achievable in the modeling tools available to the author.

Based on the results of the analysis, the standalone model of the system at hand is depicted in Figure 36.

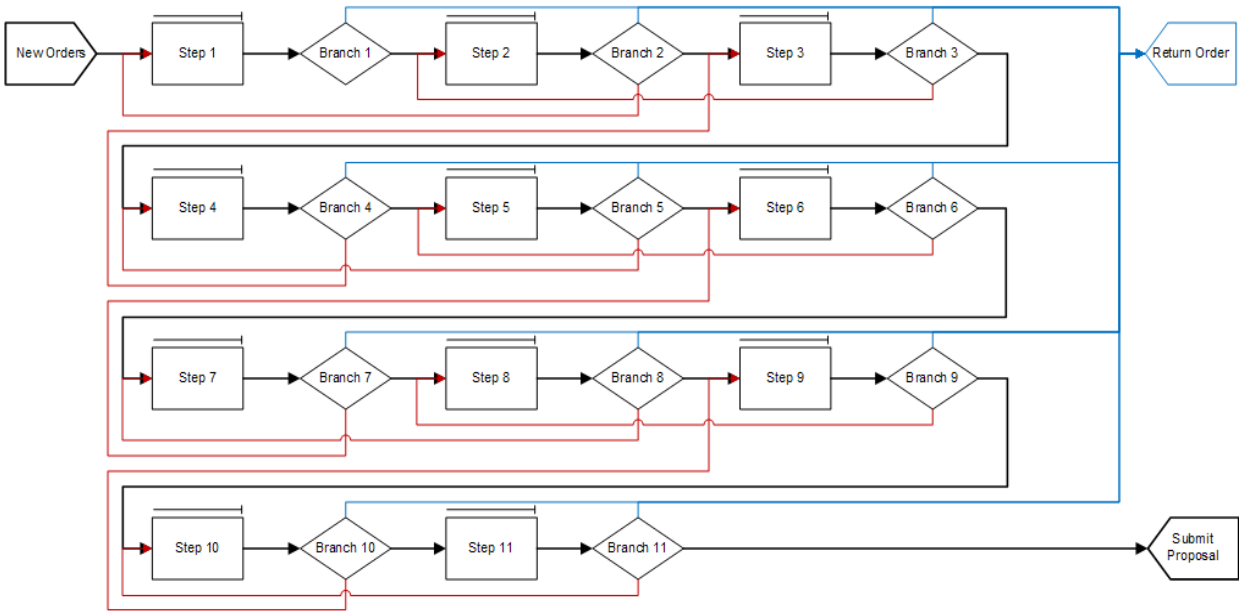


Figure 36 - Detailed DES model of system

With the prerequisites in place, the authors' prototypical scheduling subsystem to the workflow system was constructed. The diagram at Figure 37 depicts the major components of the amended workflow system.

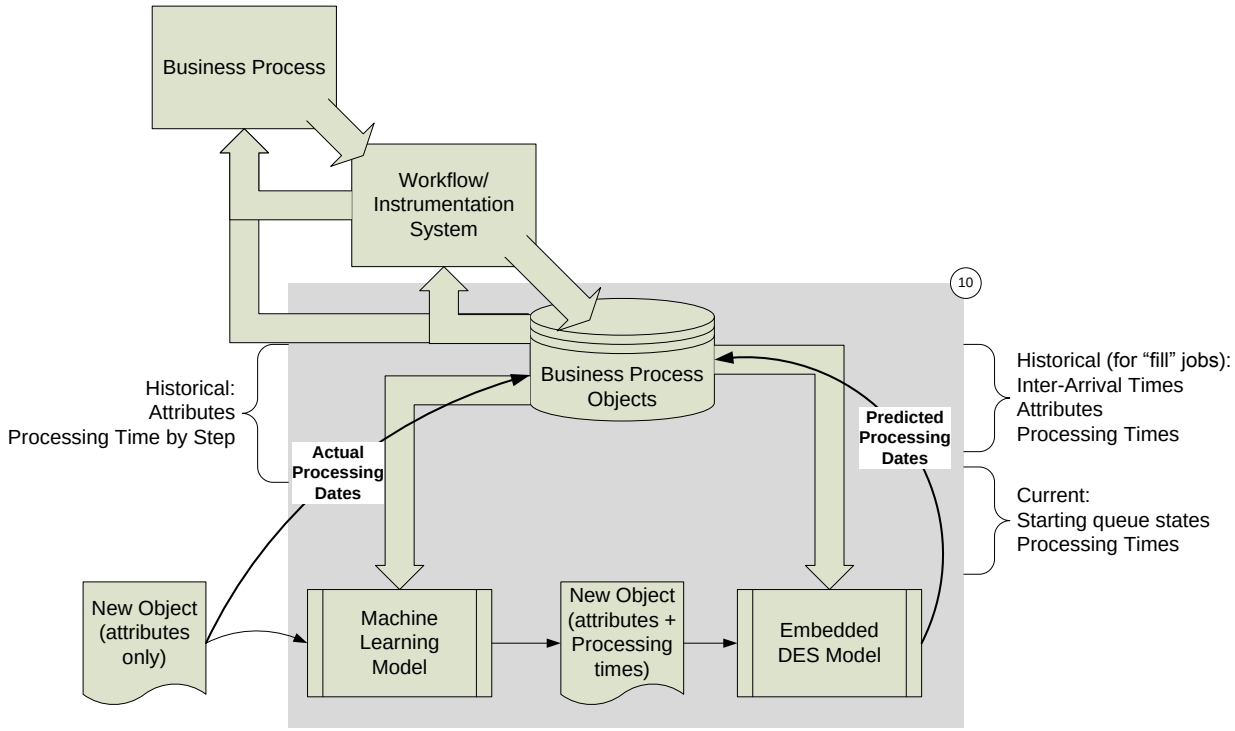


Figure 37 - Components of the amended workflow system

Mathematical Formulation

To describe the situation mathematically, consider the following definitions and relationships:

n_i: number of operations for job *i*

p_{ij}: processing time for job *i* at step *j* in its flow

w_{ij}: waiting time for job *i* at step *j*

f_i: flow time for job *i*

e_i: margin of error associated with job *i*

l_i: lead time associated with job *i*

r_i: release date for job *i*, i.e. the date that job *i* enters the system

$\hat{d}_i$: quoted due date for job *i*

d_i: actual delivery date for job *i*

L_i: Lateness of job *i* with respect to its quoted due date

q: number of jobs in process or in queue when job *i* enters the system

Assuming that there is no down time at the steps and that there is no transportation time between steps, then the flow time for a job, f_i, is simply the sum of the expected processing times for the steps for that job, p_{ij}, and the expected waiting time per step for that job, w_{ij}.

$$\boldsymbol{f_i = \sum_{j=1}^{n_i}(p_{ij} + w_{ij})} \quad (7.1)$$

Then the lead time, l_i, used to quote a due date for that job is the flow time, f_i, plus some margin of error, e_i, associated with the estimation of the processing and waiting times.

$$\boldsymbol{l_i = f_i + e_i} \quad (7.2)$$

The predicted due date for the job, $\hat{d}_i$, is then the release date for the job into the system, r_i, plus the estimated lead time, l_i.

$$\boldsymbol{\hat{d}_i = r_i + l_i} \quad (7.3)$$

Refactoring this formulation as shown below allows for a more straightforward segregation of data elements that are required for due date quoting based on the source and uncertainty of the data. To wit: the release date is given, the processing times are drawn for an

appropriate distribution, the error may be assumed or estimated from historical performance, and the waiting times are related to the jobs in queue and queuing behavior.

$$\widehat{\boldsymbol{d}}_{\boldsymbol{i}} = \boldsymbol{r}_{\boldsymbol{i}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{p}_{\boldsymbol{ij}} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij}} + \boldsymbol{e}_{\boldsymbol{i}} \quad (7.4)$$

The following relationship summarizes the salient difficulty in predicting turn-around times (TATs) in a system with non-standard queuing behavior.

$$\boldsymbol{W}_{\boldsymbol{i}} \equiv \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \boldsymbol{w}_{\boldsymbol{ij=}}\, \boldsymbol{f}\left(\boldsymbol{IAT}, \overrightarrow{\boldsymbol{P}}, \overrightarrow{\boldsymbol{Q}}, \overrightarrow{\boldsymbol{R}}\right) \quad (7.5)$$

Where IAT is the inter-arrival time for jobs that appear after job *i* arrives, and $\overrightarrow{P}, \vec{Q}, \vec{R}$ are the vectors of processing times, queuing behaviors, and rework rates respectively for the other jobs in the system. Note that the arrival process need not be stationary, and in fact, is not in the subject system [11].

Completing the formulation, the lateness of a job, L_i, with respect to its quoted due date is simply the difference between the actual delivery date, d_i, and the quoted due date, $\hat{d}_i$.

$$\boldsymbol{L}_{\boldsymbol{i}} = \boldsymbol{d}_{\boldsymbol{i}} - \widehat{\boldsymbol{d}}_{\boldsymbol{i}} \quad (7.6)$$

The author's proposed solution to determining W_i is then to (1) construct an embedded DES model, (2) determine the parameters for that model applicable at the point in time where job *i* enters the system, (3) determine the properties of job *i* necessary for representation within the model, (4) to repeatedly execute the model until an acceptable margin of error on predicting its time in system can be achieved, and (5) adjust the predicted due date based on the error distribution observed from previously scheduled jobs. With this methodology instantiated against a workflow system, the practitioner may readily answer the relevant

question: “Given a new order today, when can I expect to receive the corresponding proposal (with 90% confidence)?”

Related Literature

Cheng and Gupta [14] survey the existing research with respect to due date determination. In this survey, Cheng and Gupta [14] open by pointing out that meeting due dates is extremely important to practicing managers due to the customer service implications. They then utilize a classification scheme first proposed by Elion [15], which has six dimensions: (1) Static vs. Dynamic, (2) Deterministic vs. Stochastic, (3) Single-product vs. Multi-product, (4) Single-processor vs. Multi-processor, (5) Theoretical vs. Practical, and (6) Exogenous due dates vs. Endogenous due dates. Since exogenous due dates obviate due date quoting and lead directly to sequencing and scheduling problems, Cheng and Gupta [14] focus their attention on endogenous due dates. Using the above classification scheme, they conclude that there is very little extant research on dynamic, complex, multi-processor systems.

Subsequent to the survey conducted by Cheng and Gupta [14], Cheng [17] describes a sequencing algorithm when using the slack due date quoting policy. He simplifies the system under consideration by assuming that once a set of jobs is sequenced, no subsequent jobs will affect the system’s performance, there will be no re-sequencing of the jobs between stations and all of the earliness and tardiness costs are constant. In effect, the lack of consideration of dynamic arrival of jobs and non-permutation scheduling becomes a presupposition of first

come, first serve (FCFS). Cheng [17] concludes that an shortest processing time (SPT) sequence is optimal, although this conclusion does not fully support the findings of Duenyas and Hopp [18], who propose an analytical framework for evaluation of various job sequencing rules given that flow times can be optimally predicted. Working through a series of increasingly generalized scenarios, they conclude that an earliest due date (EDD) sequence is optimal if the tardiness penalty is constant for all customers and proportional to the tardiness, which seems to contradict Cheng [17]. To achieve this result Duenyas and Hopp [18], only assume that preemption does not take place.

Similar to Duenyas and Hopp [18], Lawrence [19] presupposes that the practitioner either has a simple system with closed-form flow time estimates, or has a method to determine flow time for complex systems. With that as a precondition, he describes an analytical approach to setting due dates based on previously-observed forecasting errors. While Lawrence [19] proposes to fit the forecasting errors, which he refers to as "*G*", using a Ramberg-Schmeiser distribution, he concludes that Erlang and Gaussian distributions worked equally well. He makes a key observation that is particularly germane in this context. Various measures of performance lead to differing uses of the error distribution. For example, mean absolute lateness is minimized by adding the median of the error distribution to the predicted flow time. Mean squared lateness is minimized by adding the mean of the distribution to the predicted flow time, and service level matching is met by adding the target percentile of the distribution to the predicted flow time, e.g., $G^{-1}(0.9)$ for a 90% service level.

Van Ooijen and Bertrand [20] introduce a distinction in terminology intended to allow some leeway between the tightly-estimated Internal Due Date (IDD) and the slightly looser External Due Date (XDD). The difference between the two is analogous to a margin of error e_i, Alfieri's Safety Time, or Lawrence's *G*, the authors propose to adjust the XDD using the ratio of the current level of work in progress (acwip) to the average level of work in progress (nwip). The results of Van Ooijen and Bertrand [20] bring some closure to the disagreement between Cheng [17] and Duenyas and Hopp [18] by noting that when earliness and lateness penalties are of similar magnitude, then SPT sequencing works best; however, when tardiness penalties are much larger than earliness costs, a due date sequencing rule is best. Another interesting observation that can be made from the data is that, in spite of the dependence on FCFS sequencing in much of the literature, FCFS is among the worst performers of the sequencing rules tested. It does, however, provide the best predictions of performance.

Much of the existing literature discusses using models of systems to conduct experiments, where the objective is to improve system performance by adjusting resources or queuing behavior [22, 23]. There is some literature that seeks to use the model to evaluate differing courses of action such as selecting a sequence of jobs to be scheduled. For example, Azzaro-Pantel, Bernal-Haro et al. [24] describe using a combination of discrete-event simulation and a genetic algorithm to optimally dispatch tasks in a job shop environment, with the genetic algorithm generating the sequences and the DES model evaluating each sequence. In a related fashion, Reijers [25] discusses using short-term simulations coupled with workflow to provide

decision support, i.e., scheduling additional resources during peak loads. Much less of the literature discusses the potential for use of the faithful model to make predictions about the system just the way it is. Rojanapibul and Pichitlamken [26] make some excellent observations about using embedded simulations to calculate prediction intervals in a flow shop environment. Cates and Mollaghasemi [27] describe the use of simulation to predict project completion dates and thereby enhance visibility of risk to better manage completion of complex projects. In both of these cases, though, the job parameters are reasonably established before the predictions are made.

This review of the literature illustrate the bounds of the current literature and highlight the lack of coverage for due date quoting in systems (in research and in practice) that do not implement rigid queuing disciplines and where job preemption and job recirculation is permissible and commonplace.

Necessity of A Novel Approach

As mentioned in the introduction, the author asserts that better predictive performance in quoting due dates should be achieved by making a faithful model of the system into which a new job is then introduced. The motivation for doing so, as well as the argument to support this assertion follows in two parts: modeling versus deterministic assessment and real-world versus ideal queuing behavior [11].

Necessity of Modeling

Meeting promised due dates is critical to customer satisfaction [14, 18, 19, 21].

Promised due dates are readily met when arbitrarily long lead times are set. However, quoting arbitrarily long lead times to ensure service levels dilutes our customer appeal while overly optimistic lead times erodes customer confidence [16]. Based on this, more accurate due dates (with narrower confidence intervals) are better (more pleasing to customers) as long as the mechanism is practical to implement [14].

As expressed in the Problem Formulation section, the due-date for a job is dependent on that job's processing times and waiting times, and should also include some safety margin [16, 17, 19].

Also from the Problem Formulation section, the dominant feature of the due-date setting problem is estimating the wait time for a given job [14].

The wait times for a job are obviously dependent on the jobs already in the system, though the particular relationship is also dependent on the queuing scheme assumed [16, 18, 19].

Including more information about the current state of the system leads to better predictions of due dates [14, 16, 18-21].

Analytical methods are suitable for simple cases with ideal assumptions, but more complicated systems require more complicated analysis typically involving simulation [14, 16, 18].

A detailed discrete event simulation model of the actual system will allow more information on the system (design, historical performance, and current state) to be brought to bear on the estimation of waiting times.

Necessity of Real-world Queuing Behavior

The data observed from the subject system for this author's research exhibits job insertion at head of line preemptively, head of line without preemption, tail of line, and other locations in the middle of the queue as depicted in Figure 38.

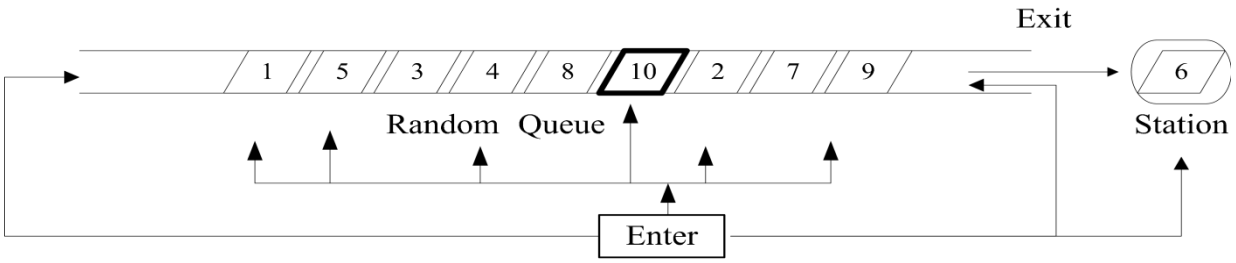


Figure 38 - Flexible Queue

Since the insertion location for a given job determines the minimum number of jobs that will be processed before that job, it provides a lower bound for the wait time of the target job at that step, but this determination is not complete, as subsequent jobs may arrive after the job in question and be queued in front of the target job increasing its wait time at that step.

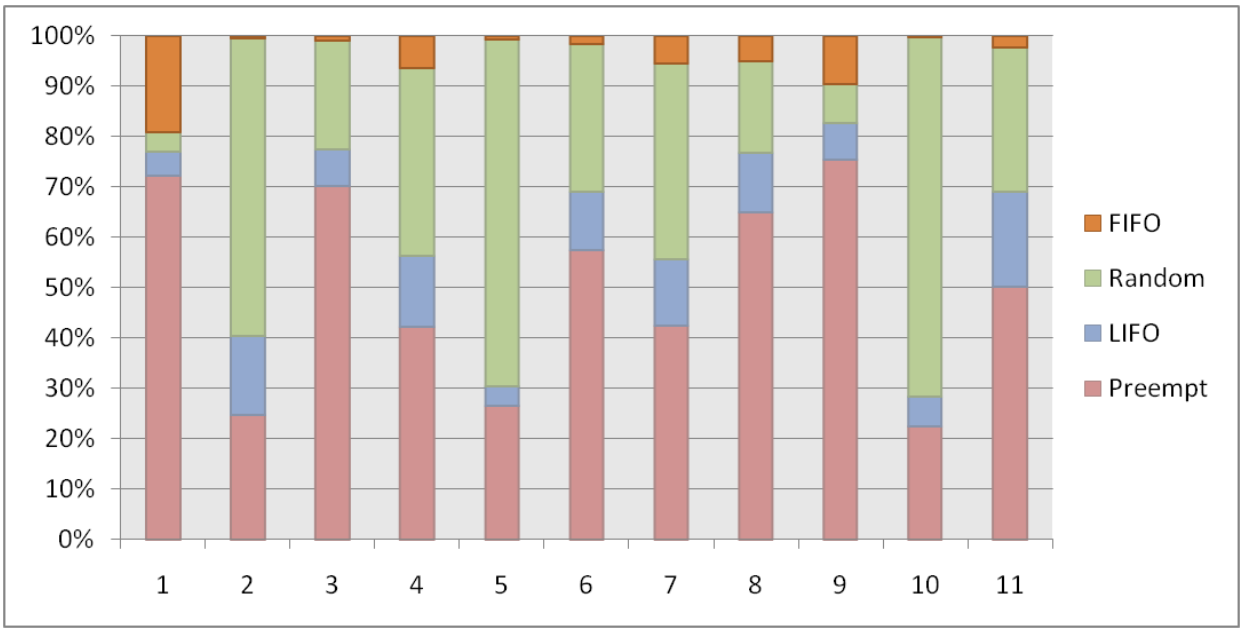


Figure 39 - Relative percentage of jobs inserted into queues by position

As mentioned in the Problem Formulation section, several thousand historical transactions are available for analysis of the system under test. By decomposing the transactions into corresponding arrival and departure events and then processing those events in departure order it is possible to glean the relative insertion position of jobs at each step. The results of this analysis are applied to the model of the system under test for this paper and expressed as the relative frequency of job insertion location by step as shown in Figure 39. These relative frequencies will be used in the empirical queuing implementation described in the "System Under Test" section. While all of the existing queuing models provide equivalent, average, system-level performance prediction, the author's goal is to accurately model the

behavior of a single, discrete job within the context of its fellow jobs, and therefore a more flexible model is required.

Argument Summation

In summary, more accurate assignment of due dates will make customers more likely to continue to place their orders using the system. Outside of certain idealized systems, incorporating more detail in the prediction process can make those predictions more accurate. A DES model allows for incorporating more system detail than any of the existing mechanisms and incorporating real-world queuing behavior is a key aspect of that mechanism. It is, therefore, worthwhile to study the forecasting performance of a faithful DES model against existing, deterministic policies [11].

Methodology

The author's prototype solution for implementing this methodology is composed of two distinct, but closely inter-related components. The first component performs an automated analysis of historical data to determine descriptive parameters for a discrete event simulation. The second component is an embedded simulation model that makes use of these descriptive parameters to replicate the behavior of the target system. It is important to note that the predictive power of this construct is dependent on both components, which must act in concert.

Automated Analysis

The automated analysis component performs five major functions: (1) decompose the departure transactions (by job and by station) from the workflow system into Departure and Arrival events, (2) use the correlated Departure and Arrival events to determine the rework rate of the sample of jobs by station, (3) use the correlated events by station, to determine the queuing behavior for that station, (4) use the correlated events by station, to decompose the total time at a station for a job into waiting time and processing time and fit the processing times to a valid statistical distribution, and (5) utilize the transaction logs, to determine the inter-arrival rate per month. The last four functions output their results to a database as a series of parameters to be used by the embedded simulation.

The first function is a pre-processing step facilitating the remaining functions. As mentioned, the system in question is an electronic workflow system. As such, there is no perceptible transportation delay. Without transportation delay, the decomposition of the departure transactions simply requires the creation of a departure event from the current station, and an arrival event at the next station visited by the job. The times of occurrence for each of these events are identical; the only complicated aspect is determining the next station visited. As this complication is purely self-inflicted by the author's implementation of transactions, recording the details of overcoming this particular hurdle will be glossed over. A sage practitioner would be well served to capture both the source and destination stations within the departure transaction and thus avoid this step entirely. As the output of this step is

only used as the input for the subsequent three steps, there is no need to store these results back to the database.

The second function uses the correlated departure and arrival events created by the first function to determine rework rates. This is accomplished simply by implementing a two-level, nested, case construct which takes at the outer-level the source station, and at the inner-level the destination station. The rework status per job is then captured as a logical action, in the author's case a job is accepted, rejected or returned without further action. The relative frequencies of these actions are recorded by station as model parameters in the database and are used by the branch components to correctly route jobs from one station to the next – this pairing of analytical and simulation components directly addresses $\vec{R}$ from Equation 7.5.

The third function, determining the queuing behavior, is considerably more interesting to describe, and is in fact, half of the novel aspect of the author's formulation for attacking $\vec{Q}$ in Equation 7.5. In general terms, the concept of the function is similar to executing a DES in reverse. In a normal DES, both the processing time for a job, and the queuing policy for a station are specified and the result for the job is the departure time from the station. In this case, however, the arrival and departure times are known and the results of the analysis are the processing time for the job, and the queuing behavior of the station. More specifically, the historical jobs arriving at a given station are processed in time-order of their arrival at the station but the jobs are placed in the queue based on their, known a priori, departure time.

Executing this process one input job at a time, it is possible to determine the queue insertion location at the station, and the accumulated processing time for the job.

For details of this process, including pseudo-code for implementation, see [28]. The concept is represented graphically in Figure 40.

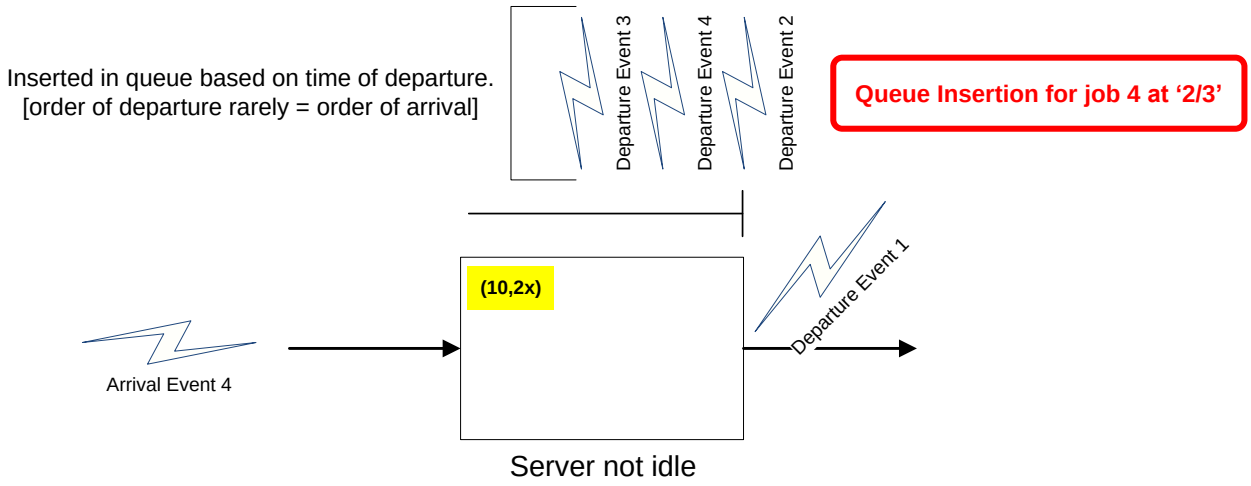


Figure 40 - Queue position determination

In pseudo-code, the virtual Server performs the following top-level tasks:

```
    Read previous 180 days of Transactions for Server;
    Create Arrival Events and Departure Events based on
transactions for completed jobs;
    loop through events in time order {
       if (arrival event) Push(event);
       else if (departure event) Pop(event);
    }
```

The pseudo-code above references 180 days of transactions as the look-back window which is appropriate in the author's business environment. Depending on the circumstances of

the practitioner's environment the look-back window might be appropriately specified in terms of days, or in terms of a number of transactions.

The output of this function, which is accomplished by the "Push" method of the virtual server, is three parameters per station specifying the fraction of jobs that preempt, queue at the head-of-line, and queue at the tail-of-line. Jobs that do not meet any of the three criteria are assumed to be randomly placed in the queue between head-of-line and tail-of-line.

The fourth function separates the processing time from the waiting time and then fits the processing times to a statistical distribution. This statistical distribution addresses, in conjunction with the server simulation component, the $\vec{P}$ component from Equation 7.5. In the author's implementation, the first portion of this function – separating processing and waiting times for a job at a station – is accomplished by a combination of the "Push" and "Pop" virtual server methods described above.

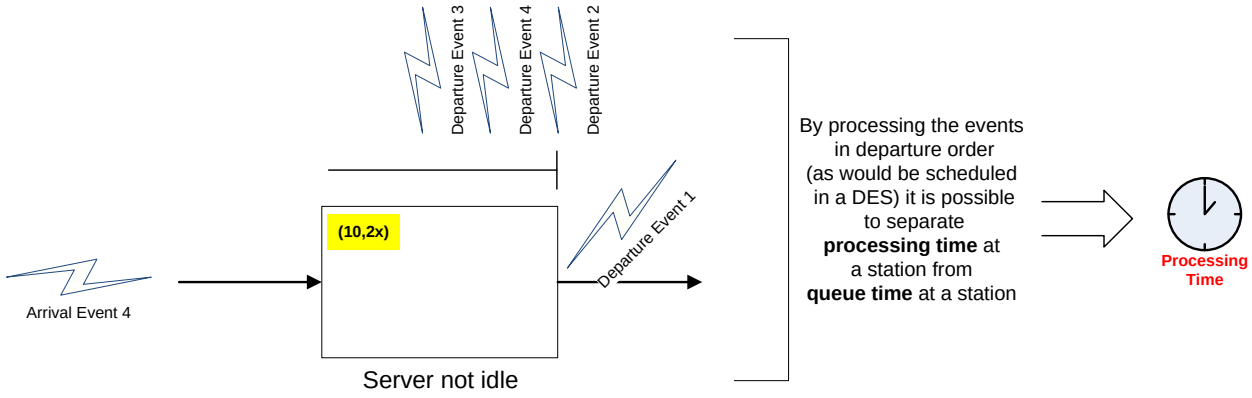


Figure 41 - Processing time determination

The second portion of the function uses a well known formulation to convolve the resulting processing times at a given station such that a linear, least-squares regression of the

convolved data exhibits the shape and scale parameters of a Weibull distribution fitted to the unprocessed data. Similar to the implementation(s) above, the newly calculated parameters are combined using exponential smoothing – as in the second and third functions – with the existing parameter values and the resultant, smoothed values stored back into the database, two parameters per station. In addition, a Kolmogorov-Smirnov goodness of fit test is executed between the source data and the fitted distribution, and the newly calculated parameters are only combined with the existing parameters if the test statistic is less than the adjusted critical value for the sample size [22].

As the reader may have already surmised, the fifth function, calculating the inter-arrival rates by month, when coupled with the source component of the simulation, completes the input parameters to Equation 7.5, namely IAT. This function is executed very simply using an SQL query which aggregates the arrivals by month for the previous 12 months. The more interesting aspects of this function reside in the simulation component discussed below.

Simulation Components

To build the embedded model used to simulate the workflow system, a series of lower-level modeling components had to be written in Java. They are described below, and shown with their key parameters in Figure 42 at the bottom of this section.

The Source component uses parameters from the database to implement a non-stationary, Poisson arrival process which varies month-by-month. At each arrival event the

Factory Component (see below) is used to generate an order entity which is sent to the output component of the source which would normally be either a Branch or a Server.

The Factory component produces, on demand, entities of type Order with processing times per step drawn from Weibull distributions whose parameters are taken from the analytical component. The Factory is also capable of creating a special "target" Order.

The Order component extends the Entity class and implements the Comparable interface. It also contains a Properties object that is used to capture the history of the event as it traverses the model.

The Server component, in conjunction with its Queue, implements the empirical queuing behavior specified by the parameters from the analytical component.

The Queue component utilizes the CompareTo() method of the Order entities to queue the Orders based on the value set for the Order by the Queuing Behavior method of the server.

The Branch component implements routing of incoming Orders to one of two or more destinations based on the rework parameters from the analytical component. The author's implementation adds special treatment for the "target" Order – it is not allowed to exit through the "return without further action" sink.

The Sink component disposes of non-target Orders as they depart the simulation, and stores the target Orders in a static collection when they exit. The Sink also signals a SimulationEnd event when the target Order exits.

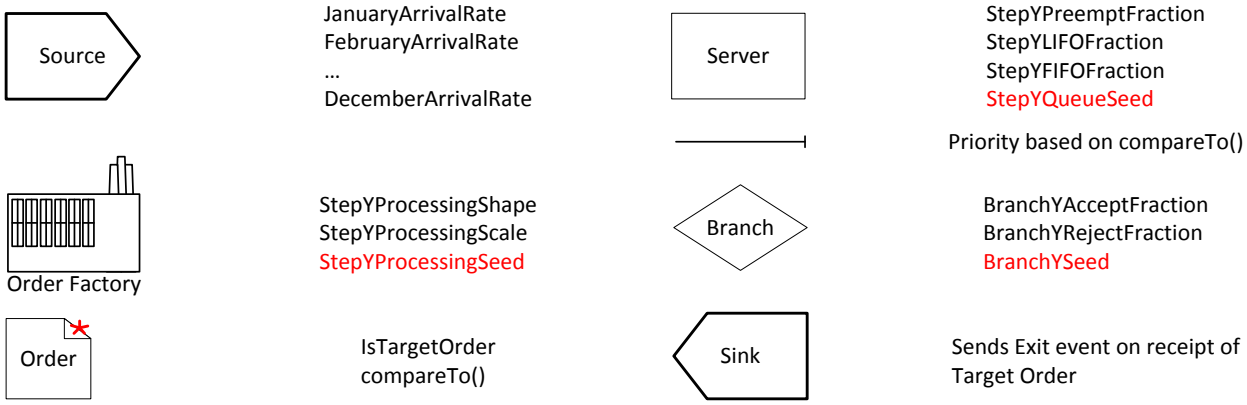


Figure 42 - Modeling components

Embedded Simulation

The top-level Java process which implements the simulation first connects to the workflow system's database. This connection is used to (1) read in the parameters generated by the analytical functions above, and (2) to determine the current state of the workflow system. A graphical representation of the embedded model is shown in Figure 43.

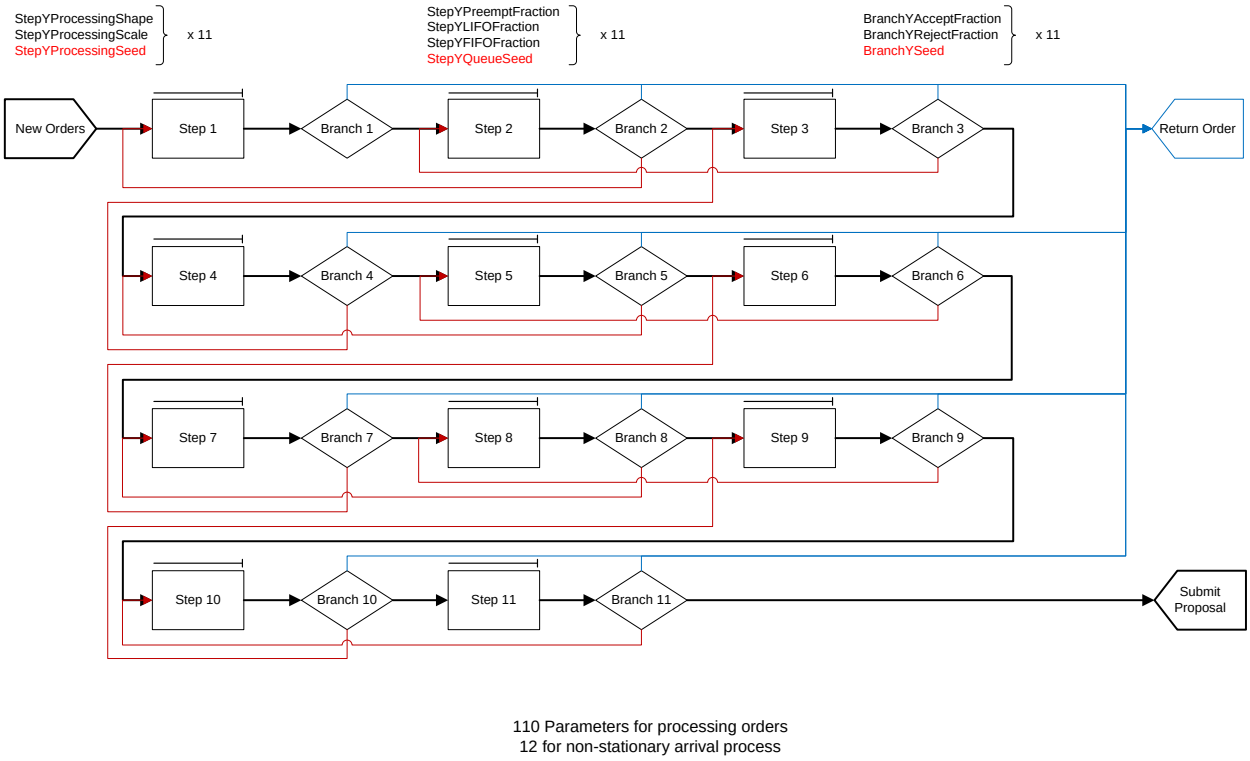


Figure 43 - Embedded model with parameters

The top-level process then instantiates the required types and quantities of modeling components using the analytical parameters. The instantiated modeling objects are then connected to each other using member functions that allow for the efficient execution of the event driven simulation. The objects are then initialized with the current state of the workflow system. At this point, the new, target order is created and enters the simulation at the first station and the simulation clock is started. The simulation runs until the target order exits the system at which point the target order and its history are added to an array of results. For multiple replications, the objects may be re-initialized (which does not reset their random number streams), a new target Order created, and the simulation again run until completion.

After the desired number of replications has been executed, statistics may be drawn from the set of resulting target Orders.

The final modification to the predictive subsystem was to incorporate the error distribution in the predictive process. After the second Order is completed, the final term in the expression for the predicted due date, e_i, can be included in the predictive process. Based on Lawrence's formulation, the target percentile of the observed error distribution for the desired service level is added to the modeled flow time [19]. If the number of completed Orders is too low, care must be taken when calculating the target percentile. If however, there are sufficient Orders completed to justify the assumption of normality (both $pn \geq 4$, and $qn \geq 4$), then simply using the product of the standard deviation of the errors and an appropriate z-value is sufficient. In this case, given a sample size greater than 40, a z-value of 1.285 (corresponding to a single-tailed, 90% area) multiplied by the standard deviation would be used for e_i to achieve a 90% service level.

Results

The prototype of the Predictive Subsystem described ran against its corresponding, production workflow system for 75 days. During this period, 119 orders were received, processed, and returned to the originating customer. As each order is completed, the error between the predicted and actual delivery dates is captured and the standard deviation of the newly expanded sample is re-calculated.

To assess the validity of the predictive process, the proportion of the orders completed within the predicted due dates, $\hat{p}$ (p-hat), is compared to the target proportion, p_0 (p-zero, which is 0.9 in this case). Instead of simply calculating the statistic for a single point in time, the authors took a time-series approach to the analysis by calculating a critical value of $\hat{p}$ based on the sample size. In the figure below, the results of this time series approach are shown. The series labeled p-hat is the observed proportion of orders that are delivered on or before the predicted due-date. The $\widehat{p_c}$ (p-hat-critical) series graphically depicts the lower bound for an observed value of $\hat{p}$ that would be statistically indistinguishable from p_0.

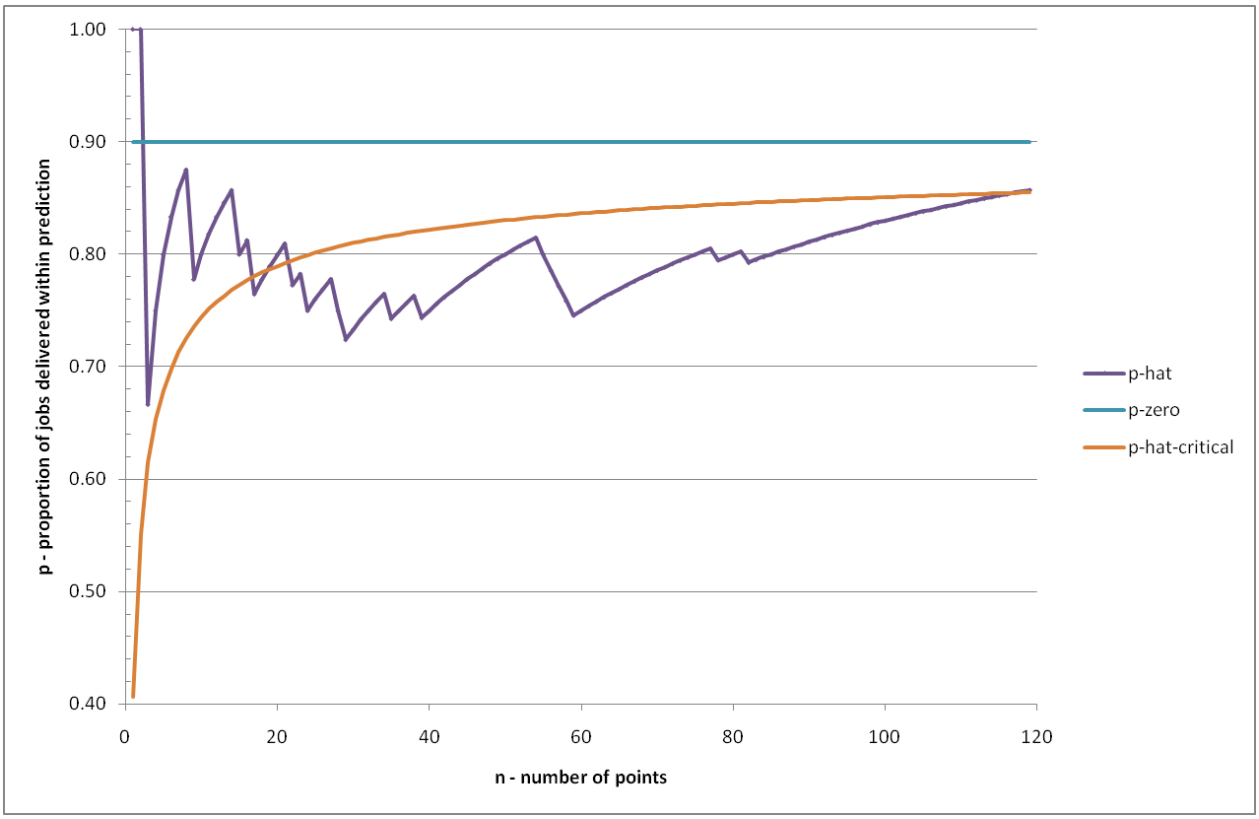


Figure 44 - Predictive performance versus job history, $P_0 = 0.90$

The p-hat-critical series is calculated as $\widehat{p_c} = p_0 - \sqrt{\frac{p_0 q_0}{n}}$.

Based on the 119 complete data points available, the authors verified Lawrence's formulation for targeted service levels between 55% and 90% in 5% increments. In each case, the value of $\hat{p}$ ended above the $\widehat{p_c}$ value indicating that the observed service level is indistinguishable from the targeted service level. The following table lists the z-values corresponding to the p0 values and the associated figures depicting the results.

Table 4 - Result figures by P_0

P_0	Z value	Figure
0.55	0.125	Figure 45
0.60	0.253	Figure 46
0.65	0.390	Figure 47
0.70	0.525	Figure 48
0.75	0.675	Figure 49
0.80	0.841	Figure 50
0.85	1.036	Figure 51
0.90	1.285	Figure 44

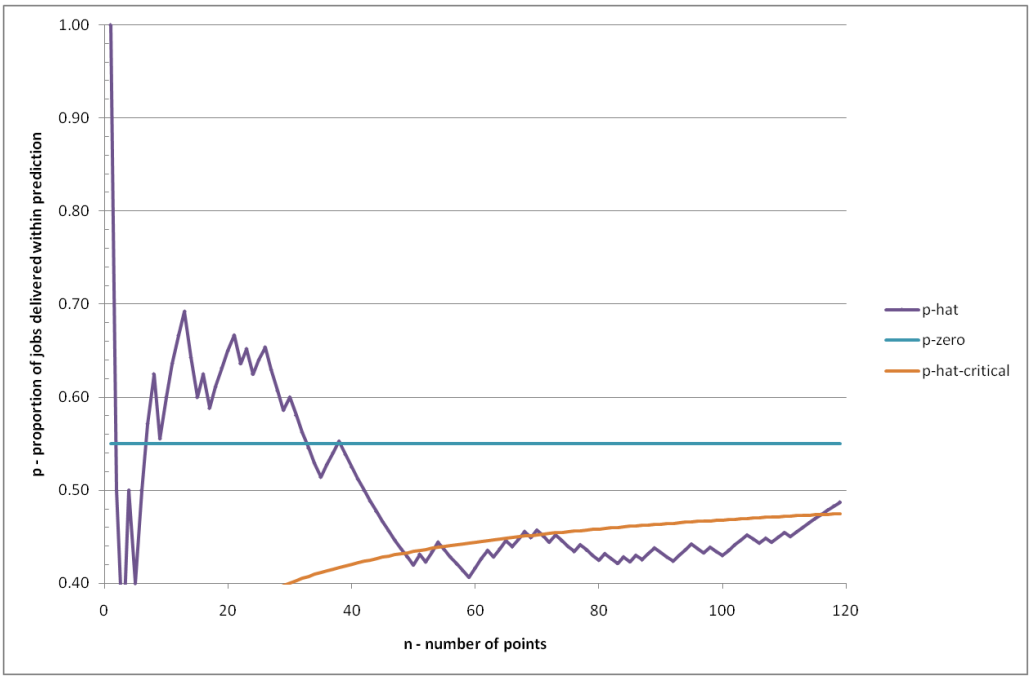


Figure 45 - Predictive performance versus job history, $P_0 = 0.55$

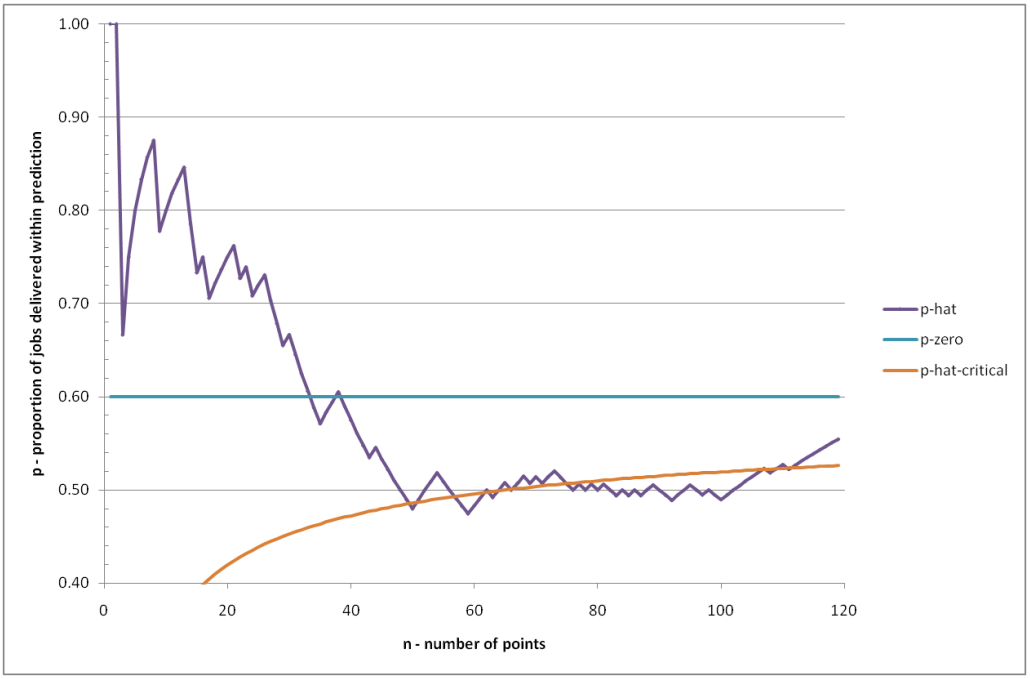


Figure 46 - Predictive performance versus job history, $P_0 = 0.60$

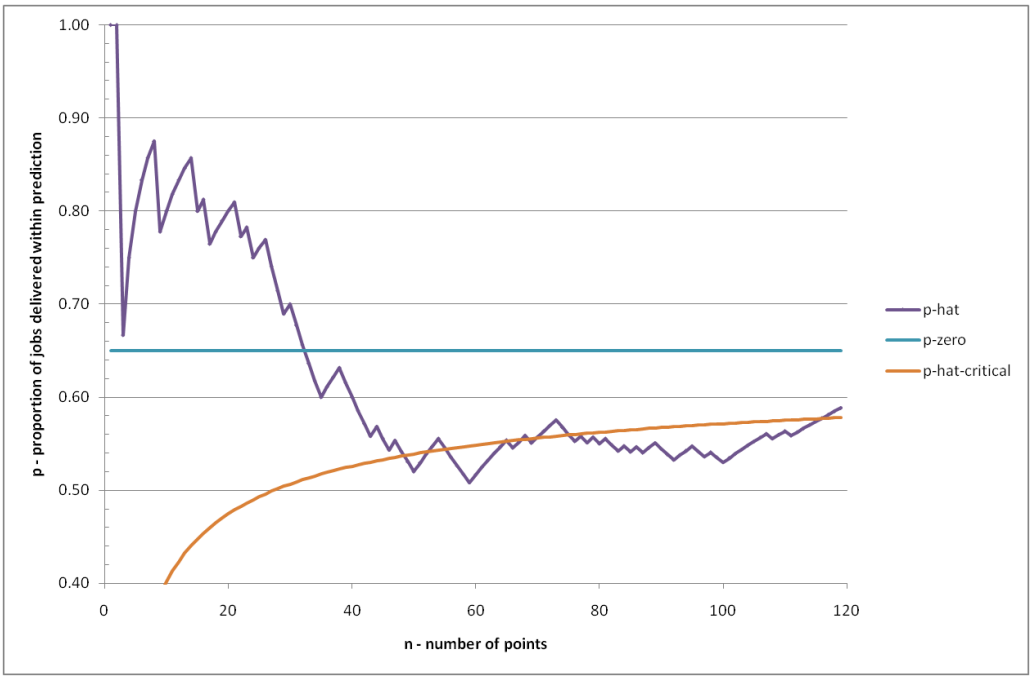


Figure 47 - Predictive performance versus job history, $P_0 = 0.65$

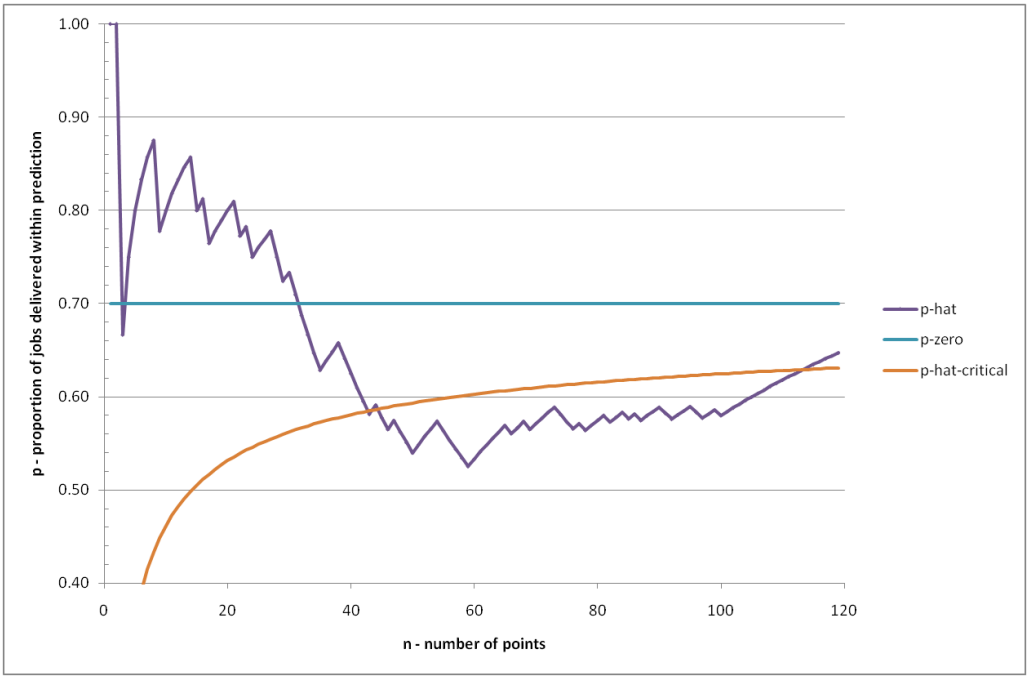


Figure 48 - Predictive performance versus job history, $P_0 = 0.70$

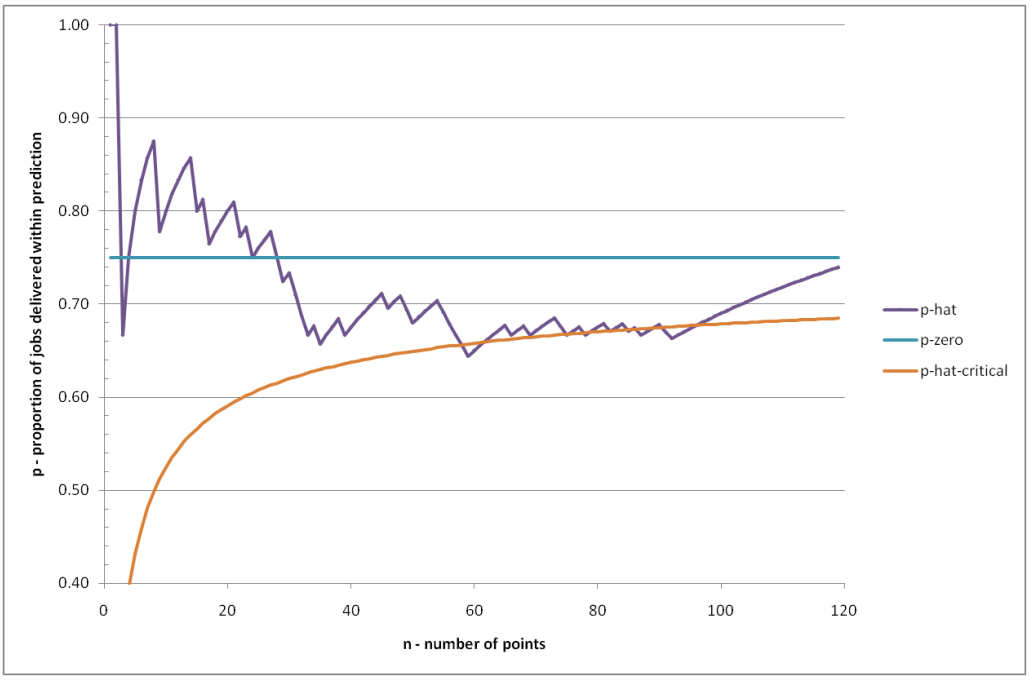


Figure 49 - Predictive performance versus job history, $P_0 = 0.75$

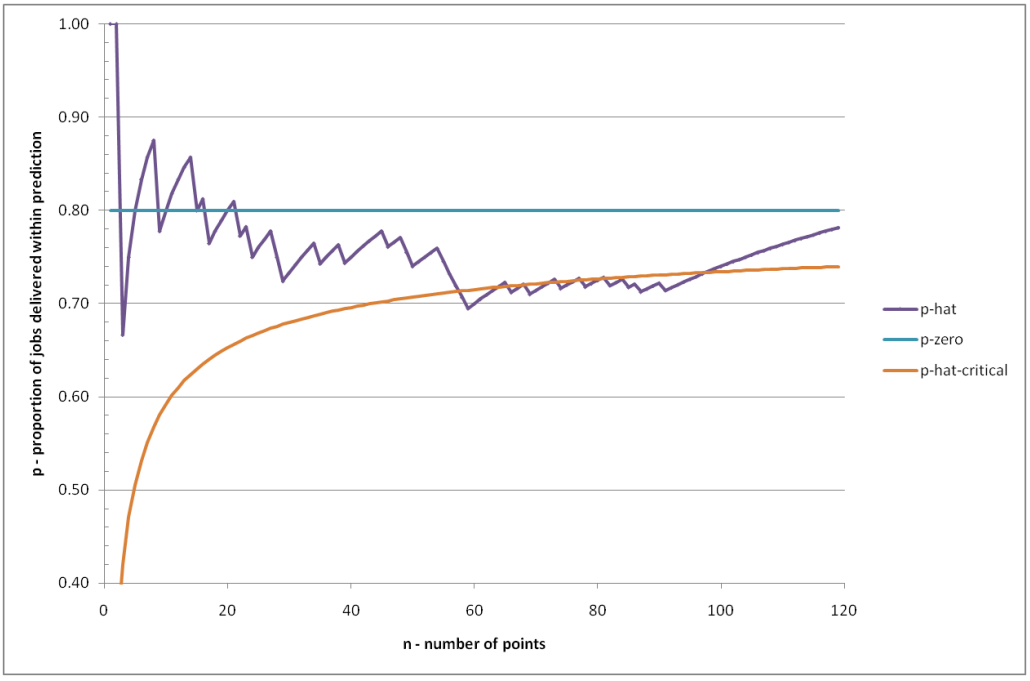


Figure 50 - Predictive performance versus job history, $P_0 = 0.80$

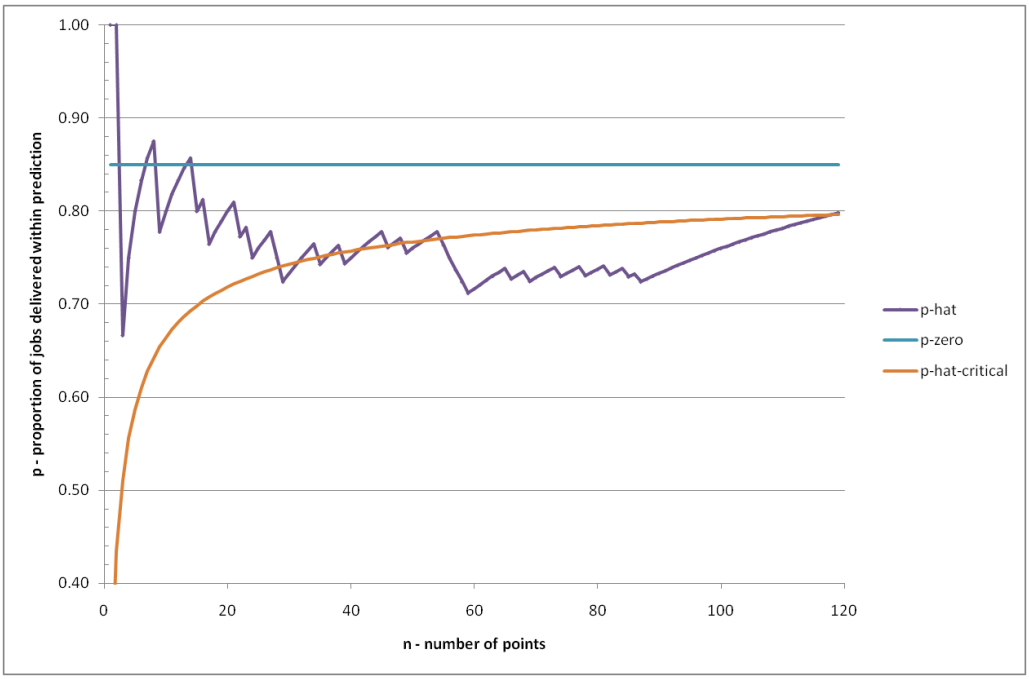


Figure 51 - Predictive performance versus job history, $P_0 = 0.85$

Discussion

As the reader may have observed, the achievement of an arbitrary service level is trivial in the endogenous due date case, i.e. if the manufacturer of a widget is allowed to determine his own due date for delivery within the bounds of some service level he may simply quote a date far enough in the future such that no readily conceivable circumstance might cause him to miss his due date - ten times the duration of the worst case scenario, for example. In practice, these extravagant delivery times tend to alienate customers. Depending on the circumstances of both the customer and the supplier, a balance must be reached between arbitrarily inflated delivery times and missed deliveries. In the case of goods, there is often a cost associated with

early delivery, such as storage – either held before delivery by the manufacturer, or stored until required by the customer – in either case, there is a measurable cost of storage space over some period of time to be accounted for.

With services however, the cost of early delivery is less tangible. There is no measurable cost associated with the storage of a simple electronic document for an additional week or two. In the services case, especially with endogenous due dates, the cost for early delivery lies in the realm of perceptions. If the service provider consistently and extravagantly overestimates the delivery date, the customer may resent paying the premium associated with a "guaranteed service level", especially when the provider appears to be padding his estimates. Compounding the problem, there is also no measurable cost for the service provider to hold on to an electronic file until the quoted due date.

The authors' approach to this dilemma is to share as much of the raw processing data and due-date quoting methodology as possible with current and prospective customers. It is only in this transparency that trust can be formed.

Conclusions and Future Research

Given the success of the prototypical implementation, future work will focus on the implementation of this methodology in a production system such that a premium may be charged for the meeting of specified service levels. As mentioned in the discussion section above, of equal importance to the implementation of the production system will be the

transparent communication of the fairness of the quoted due dates. The authors expect that continued research on this topic will lend credibility to this methodology.

CHAPTER EIGHT: CONCLUSIONS AND FURTHER RESEARCH

The due date quoting methodology proposed and implemented in this research has been effective in providing accurate delivery targets for the orders processed through the target workflow system. Since the implementation of this methodology on the production system, the delivery predictions have been met in accordance with the service level specified. In addition to simply specifying a final delivery date, the system also produces step-by-step milestones leading to the predicted due date. This capability has improved management confidence in meeting our service targets and provided the framework for efficient measurement of progress. Management of this performance is handled simply through a daily review of expected progress, expressed as the predicted milestones against the actual progress of the orders. A reporting tool was developed and deployed that produces an up to date view of pending and late steps across all of the orders in the system. The tool also allows subordinate managers and functional workers to continually monitor their progress on their orders.

Practical Implications

Within the next three to nine months the predicted dates may well become contractually binding. If and when the decision is made, it will be mutually agreed to by our customers and management. To effect this change, an additional output capability will be

activated which will send email notification to the requesting customer and the responsible manager of the new orders promised delivery date.

During the development of the workflow system that underlies this research as well as the predictive subsystem which is its subject, a series of practical considerations were collected. These considerations are predominantly concerned with human behavior, business process definition, and software usability. As these topics are important to successfully repeating the process described in Chapter Two they have been included in this document. However, as they are outside the author's academic background they have been included in an appendix (APPENDIX B: PRACTICAL CONSIDERATIONS) instead of within the body of this document. Readers wishing to implement the subject methodology are encouraged to refer to this section before beginning work.

Future Work

Performance of the modeling component will be improved through the thoughtful incorporation of multi-threading support whilst maintaining the appropriate control on the several random number streams used.

An optimization method will be incorporated into the prediction system such that the error term used in calculating the delivery date may be adjusted for unequal earliness and tardiness penalties. The preliminary implementation will allow for a system wide parameter indicating the relative value of the penalties, e.g. the tardiness penalty is five times as large as

the earliness penalty. Subsequent implementations may allow for the parameter to be set on a job-by-job basis.

The author was fortunate in having thousands of historical records upon which to predicate his analysis. If the described process were undertaken from scratch, parameters such as the exponential smoothing factor (usually denoted by α) would have to be set carefully to achieve reasonable performance until such point that sufficient jobs might consistently be available in the historical window to dilute the criticality of this parameter. Similarly, the determination of historical window sizes for the determination of queuing behavior, processing times, rework rates, and error distributions should be parameterized for other applications.

Given the strong correlation between the number of jobs in work and the processing time for a new job entering the system (see Figure 26), it may be possible to reduce the simulated complexity heuristically to a formula that relates the flow time for a job, f_i to some sort of cross product between the queuing behaviors at each server (Q), and the total processing time for all of the jobs at that server (P) such that the due date could be quoted as $d_i = r_i + \text{QxP} + \text{z}e_i$.

As postulated by Ferreira and Ferreira [29], the author will look for a suitable, standards-based, workflow framework to rehost the subject business process. However, before any rehosting can be considered, such a framework must demonstrate similar capabilities (web access, open development, extensible data structures, and clear integration points) to the developmental system described herein. If such a commercial product cannot be found, then

following through with Milainovic, et al.[30] and several of the other sources a better developmental solution may be pursued.

APPENDIX A: LITERATURE REVIEW

There exists an extensive body of research on Discrete Event Simulation, Business Process modeling, workflow systems, data mining and optimization. This review will touch on the existing literature in this area but predominantly focus on highlighting the gaps in that literature with respect to the modeling of systems with high levels of non-deterministically defined parameters and using the resultant models to make specific predictions about individual jobs as opposed to general system performance. The structure of the literature review will parallel the 10-step system development process introduced in the first chapter and each section will have a separate grid cataloging the articles.

Business Process Modeling

Discrete Event Simulation modeling has been predominantly focused on manufacturing and other production systems where the individual steps in the process are well defined and often repeated. As our society transitions away from production and towards services, the tools of the Industrial Engineer must adapt. Gladwin and Tumay stated that a business process is a collection of logically interrelated activities that consume resources to achieve specific objectives [31]. Within that context, they explored modeling business processes within simulation tools and applying those simulation tools to improve performance of business processes outside of manufacturing where such tools have been predominantly used. Of course, before such a model might be created it is necessary to capture the business process often with the intent of building an information system to support the process. Cook, Rozenblit

et al. described their use of UML diagrams to capture a desired business process management system [32]. The evaluation of business processes with the intent of improving speed or efficiency is often referred to as Business Process Re-Engineering. Bae, Jeong et al. discussed the potential linkage between a business process model and a correlated simulation as a means of analyzing the impacts of proposed changes to the business process as part of a business process re-engineering exercise [33]. Ghanmi provided a solid, real-world example of how this correlation between process and model can be drawn in a product-centric environment [34]. Similarly, Jianhua, Zhibin et al. presented a case study for a parallel situation in a more service-centric environment [35].

It is commonly considered that one of the key distinctions between a manufacturing process and a business process is that business processes may involve mechanical or technological components but is predominately a human-centric endeavor and, therefore, inherently more difficult to model than the more mechanical processes that dominate the manufacturing world. As a practical consequence of this, Gladwin and Tumay made good points about accounting for non-deterministic processing times and variable processing capacity [31].

The benefits of capturing an extended business process and creating a workflow around the process is nicely described by Abecker, Bernardi et al. and Kayser, McIntosh et al. in which they rightly concluded that this exercise, even before any more extensive changes are made to the business process, is of tremendous benefit to all of the parties involved through enhanced

communications [36, 37]. As a counterpoint to this, however, Reijers, Song et al. concluded that a collaborative workflow system is not solely sufficient to level the communications gradient across geographically distributed work forces [38].

Title	Primary Author	siness Process Modeling	orkflow Systems	ta Mining	crete Event Simulations	bedded Simulations	edictive use of Simulations
Information supply for business processes: Coupling workflow with document analysis and information retrieval	Abecker, A.	X	X				
Meta-Manager: A requirements analysis	Cook, Jay F.	X	X				
Modeling and analysis of a Canadian Forces Geomatics division workflow	Ghanmi, Ahmed	X	X				
Workflow management based on process model repositories	Gruhn, Volker	X	X				
Comparative research of modeling methods for workflow process	Jiang, Guoyin	X	X		X		
A surgical management information system driven by workflow	Jianhua, Qi	X	X				
New generation well project management application improves cycle time, workflow efficiency, corporate compliance, and knowledge sharing (and people like it!)	Kayser, H.	X	X				
Process ownership challenges in IT-enabled transformation of interorganizational business processes	Larsen, Michael Holm	X					
Modelling business processes with workflow systems: An evaluation of alternative approaches	Mentzas, Gregory	X	X				
Integrating light-weight workflow management systems within existing business environments	Muth, Peter	X	X				
A user-oriented design for business workflow systems	Pourabdollah, Amir	X	X				
Analysis of a collaborative workflow process with distributed actors	Reijers, Hajo A.	X	X				
Workflow simulation for operational decision support	Rozinat, A.	X	X		X		X
Research and implementation of workflow interoperability crossing organizations	Wan, Dingsheng	X	X				

Figure 52 - Business Process literature grid

Workflow

As a logical precursor for the workflow system's development, Jiang and Dong compared different frameworks that can be used for creating the workflow model from the business process [39]. Correspondingly, Mentzas, Halaris et al. reviewed various available workflow systems for suitability, highlighting the strengths and limitations of each [40].

Gruhn and Schneider pointed out that workflow tools and frameworks have existed for some time but that they had not been widely exploited owing, in their estimation, to the lack of building blocks from which to build reasonably complex systems. Their proposed solution was to provide a repository of such sub-process snippets which they deemed helpful in building up more complex workflows for well structured processes such as software development [41].

Ames, Burleigh et al. discussed the concept of a web-based workflow management system and provided some example applications that have been developed [42]. Liang, Wu et al. reviewed the, then extant, techniques in web-based workflow management [43]. Hong Va, Kei Shiu et al. refined this discussion by describing the potential use of CORBA as a representative "distributed object management" means of allowing for the encapsulation of distinct pieces of the process logic while facilitating interoperability [44]. As CORBA fades from the modern parlance, Jin, Wu et al. and Wan, Li et al. presented corresponding web services cases [45, 46]. As an alternative to integrating functional systems into the workflow system, Muth, Weissenfels et al. proposed what they term a light-weight workflow implementation for use within existing business automation environments [47]. This author prefers to make this

distinction ontologically by referring to such constructs as instrumentation systems. Huang made an excellent observation that the modern marketplace is replete with interwoven consumer-supplier relationships and supply chains. Huang then concluded that a distributed workflow system is required in this environment [48]. Similarly, Sayal, Casati et al. proposed that existing Business-to-Business statndards might be leveraged to that purpose [49]. This author would argue that a single *shared* workflow solution is also an acceptable solution to this problem given the requisite infrastructure and security means are available.

Botha and Eloff rightly cautioned that access control within workflow systems needs to be rooted in the underlying business process, however, they somewhat naively settle on using a purely role-based acess control (RBAC) scheme overlooking the importance of some robust, assignment based extension to that scheme [50]. Lin, Zhan et al. proposed an extension of the basic RBAC framework that would be organizationally aware [51]. Yu, Chen et al. described a multi-policy access scheme that extends RBAC by providing access controls at the object level [8]. Alternatively, Chen and Feng described an extension of RBAC that extends to a Digital Rights Management (DRM) level of granularity as a way of overcoming an RBAC system's limitations [52].

As a practical consequence of knowing which users are executing process steps within the workflow, it is possible to use the system to verify that particular actions were completed. Dallien, MacCaull et al. discussed the value of this "verifiability" in a medical context [3]. It is

always remarkable to the author that providing a person with a very precise date-time-action reference is an excellent aid to their memory.

Title	Primary Author	siness Process Modeling	orkflow Systems	ta Mining	crete Event Simulations	bedded Simulations	dictive use of Simulations
Applications of Web-based workflow	Ames, Chuck		X				
Integration of workflow management and simulation	Bae, Joon-Soo		X		X		
A framework for access control in workflow systems	Botha, Reinhardt A.		X				
Study of information security in workflow management system	Chen, Liwan		X				
Initial work in the design and development of verifiable Workflow Management Systems and some applications to health care	Dallien, Jeff		X				
Developing a reusable workflow engine	Ferreira, Diogo M. R.		X				
Web-based workflow framework with CORBA	Hong Va, Leong		X				
Distributed manufacturing execution systems: A workflow perspective	Huang, Chin-Yin		X				
Service-oriented workflow model	Jin, Yueping		X				
From semantic to object-oriented data modeling	Kilov, Haim		x				
Research of Web-based workflow management system	Liang, H.		X				
Improved RBAC model based on organization chart	Lin, Lin		X				
On some problems while writing an engine for flow control in workflow management software	Milainovic, Boris		X				
Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making	Reijers, H.A., van der Aalst, W.M.P.		X		X		X
Integrating workflow management systems with business-to-business interaction standards	Sayal, Mehmet		X				
Business process mining: An industrial application	van der Aalst, W. M. P.		X	X			
Multi-policy access control model for workflow management system	Yu, Ling		X				

Figure 53 - Workflow literature grid

Data Mining

As brought out initially in the introduction, there is a significant quantity of attribute data associated with the objects entering and flowing through the workflow system in question as well as several thousand historical performance records associated with previous objects.

van der Aalst, Reijers et al. made the excellent point that modern information systems (and specifically workflow systems) capture much of the necessary data to perform data mining on the process information, which they termed “process mining” without having to resort to external data collection though there have been few real-world exploitations of this capability captured in the literature [12]. Rozinat, Wynn et al. proposed to extend this concept through the use of a pair of open source tools -- YAWL (Yet Another Workflow Language) and ProM (Process Miner). They described the potentially tight coupling theoretically possible between a workflow system and a simulation model that represents that system. This coupling would be accomplished by describing the workflow system in YAWL, running the resultant workflow description through the YAWL runtime, and then developing plug-ins for ProM that would (1) allow it to ingest the system design and (2) interpret the transaction and state information. Rozinat successfully created an example of this coupling using a simple credit processing workflow. It is important to note Rozinat’s conclusion -- that while the concept seems valid, the creation of a generalized process for achieving coupling was not yet obtainable [13]. In addition to the limitations imposed by the developmental nature of Rozinat’s plug-ins for reading YAWL information into ProM, there are also limitations based on ProM itself in that there currently are not facilities to support the generalized queues that are necessary to support certain real-world processes such as the one under consideration.

One of the complexities associated with exploiting this process mining capability is the high dimensionality of the attributes and more specifically the dominance of nominal

dimensions over both ordinal and real. There are several approaches to this complex problem which are well represented in the literature so the author will only touch on the highlights here. Basic topics in simple and multiple regression are well covered in fundamental texts such as "Statistics for Engineering and the Sciences" [10]. These tools are a reasonable point of departure for simple datasets but rapidly become unwieldy as dimensionality grows. More advanced techniques seek to incorporate some dimensionality reduction schemes into the approach minimizing the amount interaction required by the practitioner. Given the high nominal dimensionality of the author's data two approaches will be given further treatment, Classification And Regression Trees and Artificial Neural Networks. Beginning with their original monograph, Breiman, et al. [53] described their novel approach to an automated process for dealing with high-dimensionality data sets through the use of what they called Classification And Regression Trees (CART™). While quite good at classifying data sets, CART's ability to provide accurate models where the dependent variable is real-valued is limited. Artificial Neural Networks, e.g. Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) often provide better results with real-valued outputs at the cost of a less easily understood model. Additional detail on MLPs can be found in Cybenko's work [54]. Similarly, a description of the SVM is found in Vapnik's paper [55]. An explanatory paper by Bennet and Campbell lent some clarification to the underlying principles behind the support vector [56]. While Chen, Ma, et al. provided a practical example of using SVMs to mine consumer credit card data [57].

Chiu, Tien et al. performed generalized comparative analyses of the various techniques commonly used in data mining and dimensionality reduction and proposed some interesting hybrid approaches to increasing accuracy while maintaining the ability to automate the process [58]. A similar comparison was conducted by Meyer, D., F. Leisch, et al. which compared various classification and regression techniques to SVMs and came to similar conclusions – namely that different techniques work better depending on the situation [59]. Given the array of such tools and their complex sets of strengths and weaknesses, the author has shortened his lines by choosing to employ a package called WEKA (Waikato Environment for Knowledge Analysis) which provides a generalized framework for evaluating the output of many of these algorithms against a given data set. As an added benefit WEKA will create executable Java modules which encapsulate the selected output model greatly simplifying automation [60].

Title	Primary Author	siness Process Modeling	orkflow Systems	ta Mining	crete Event Simulations	bedded Simulations	dictive use of Simulations	Heuristic	Stochastic
Performance comparison of trained multi-layer perceptrons and trained classification trees	Atlas, Les			X				X	X
Support Vector Machines: Hype or Hallelujah?	Kristin P. Bennet			X				X	X
Classification And Regression Trees	Leo Breiman			X				X	X
Mining the customer credit using hybrid support vector machine technique	Chen, Weimin			X				X	X
Construction of clustering and classification models by integrating Fuzzy ART, CART and neural network approaches	Chiu, Chih-Chou			X				X	X
Approximation by superpositions of a sigmoidal function	Cybenko, G.			X				X	X
The WEKA Data Mining Software: An Update	Mark Hall			X				X	X
The support vector machine under test	Meyer, David			X				X	X
Empirical input distributions: an alternative to standard input distributions in simulation modeling	Shanker, A.			X	X			X	X
Business process mining: An industrial application	van der Aalst, W. M. P.		X	X				X	X
Support vector method	Vapnik, V. N.			X				X	X

Figure 54 - Data Mining literature grid

Simulation, Modeling And Analysis

Modeling, especially Discrete Event Simulation (DES) modeling is a well documented field as evidenced by the availability of textbooks for teaching the subject at the undergraduate and graduate levels. Law and Kelton's "Simulation Modeling and Analysis" is a good example of such a text [22]. Since the basis of the field is well settled, the author will forego any detailed review of this segment of literature which encompasses what the author will refer to as classical DES modeling and includes queue types, processing and server types and

configurations, validation and verification of models, and distribution fitting. Detailed discussion of the author's approach to validation and verification will follow in chapter 3, but will fall generally in line with accepted techniques as covered in [9, 22].

While the central aspects of DES modeling are stable, there are aspects that continue to be refined such as the representation of observed data in some mechanism allowing for the generation of similar data. This is most often done by analyzing the observed data with some software, e.g. ExpertFit, and selecting one of the recommended distributions that closely approximates the observed data. There are times, though, that the observed data is intractable to such analysis and building an empirical distribution is a better solution [61, 62]. In this author's case it appears that it is not only the input distributions that have to be addressed, but also the queuing behavior. Normally queues are modeled as First In First Out (FIFO), Last In First Out (LIFO) or some form of priority queue [23]. This limitation could induce unacceptable errors when real-world queuing behavior is not as cleanly exhibited.

Embedded Modeling And Simulation

The creation of DES models outside of an Integrated Development Environment can be done from the ground up, however, several software frameworks are available that provide most of the infrastructure required for robust DES model development. These models can then be embedded into other systems to provide a modeling capability. Often that capability can then be used to evaluate the "goodness" of a particular job or batch sequence through a

system. As with the basic foundations of simulation, embedded DES is a well covered topic with good texts such as Garrido's "Object-Oriented Discrete-Event Simulation with Java – A Practical Introduction" as evidence [63].

Predictive Use Of DES Modeling

Much of the existing literature talks about using models of systems to conduct experiments where the objective is to optimize system performance by adjusting resources or queuing behavior [22, 23].

There is some literature that seeks to use the model to evaluate differing courses of action such as selecting a sequence of jobs to be scheduled. For example, Azzaro-Pantel, Bernal-Haro et al. described using a combination of a discrete event simulation and a genetic algorithm to optimally dispatch tasks in a job shop environment, with the genetic algorithm generating the sequences and the DES model evaluating each sequence [24]. In a related fashion, Reijers discussed using short-term simulations coupled with work flow to provide decision support, i.e. scheduling additional resources during peak loads [25]. And as mentioned in the Data Mining section above, Rozinat, Wynn et al., as part of their work with YAWL and ProM, described a methodology that should allow for the creation of the model from the description and output of the system and then using that simulation to make decisions about the system. With the coupled simulation model created and loaded it should then be possible to use the simulation

to answer system level performance questions and conduct "what-if" experiments to evaluate changes in resource levels that might affect overall system performance [13].

Much less of the literature discusses the potential for use of the faithful model to make predictions about the system *just the way it is*. Rojanapibul and Pichitlamken made some excellent observations about using embedded simulations to calculate prediction intervals in a flow shop environment [26]. Cates and Mollaghasemi described the use of simulation to predict project completion dates and thereby enhance visibility of risk to better manage completion of complex projects [27]. In both of these cases, though, the job parameters were reasonably established before predictions were made.

Title	Primary Author	siness Process Modeling	orkflow Systems	ta Mining	crete Event Simulations	bedded Simulations	dictive use of Simulations	Heuristic	Stochastic	Real time	Preprocess	Optimization	Prediction	Processing times known *a priori* randomly selected	Processing times and euing behavior inferred
A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm	Azzaro-Pantel, Catherine				X	X	X	X	X		X	X		X	
Integration of workflow management and simulation	Bae, Joon-Soo		X		X				X		X	X		X	
Object-oriented discrete-event simulation with Java: a practical introduction	José M. Garrido				X	X		X	X						
Empirical discrete distributions in queueing models	Jewkes, Elizabeth M.				X			X	X						
Comparative research of modeling methods for workflow process	Jiang, Guoyin	X	X		X			X	X		X	X		X	
Simulation with Arena	W. David Kelton				X				X		X	X		X	
Simulation Modeling and Analysis	Averill M. Law				X			X	X						
Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making	Reijers, H.A., van der Aalst, W.M.P.		X		X		X		X	X		X		X	
Workflow simulation for operational decision support	Rozinat, A.	X	X		X		X	X	X	X		X	X	X	
Empirical input distributions: an alternative to standard input distributions in simulation modeling	Shanker, A.			X	X			X	X						

Figure 55 - Simulation literature grid

Due Date Quoting

Cheng and Gupta [14] produced a survey of the existing research with respect to due date determination. In this survey, Cheng and Gupta opened by pointing out that meeting due dates is extremely important to practicing managers. They then utilized a classification scheme first proposed by Elion [15] which has six (6) dimensions: (1) Static versus Dynamic, (2) Deterministic versus Stochastic, (3) Single-product versus Multi-product, (4) Single-processor versus Multi-processor, (5) Theoretical versus Practical, and (6) Exogenous due-dates versus Endogenous due-dates. Since exogenous due-dates obviate due-date quoting and lead directly to sequencing and scheduling problems, Cheng and Gupta focused their attention on endogenous due-dates. Using the above classification scheme they concluded that there is very little extant research on Dynamic, Complex, Multi-processor systems. And after noting that better predictors would be beneficial, if practical, they concluded that there is a need for more practical and applied research in this area.

Alfieri [16] proposed two new quoting policies based on setting a static Safety Time (ST) parameter analogous to e_i in the formulation from Chapter Three noting that setting this parameter dynamically could be time consuming. The performance of these quoting policies, which both presuppose a First-Come-First-Served (FCFS) ordering, is compared to the Total Work Content (TWK) policy when jobs are sequenced by Shortest Processing Time (SPT), Earliest Due Date (EDD) and First-In-First-Out (FIFO). These comparisons were predicated on batch scheduling (ignoring subsequent arrivals), deterministic processing times and non-

permutation sequencing. With these simplifications, her results indicated that TWK outperforms both of her proposed policies. She noted that estimating flow times for more complicated systems is a suitable topic for future research.

Subsequent to the survey conducted with Gupta discussed above, Cheng [17] described an efficient and optimal sequencing algorithm when using the slack due-date quoting policy. Cheng simplified the system under consideration by assuming that once a set of jobs is sequenced, no subsequent jobs will affect the systems performance, there will be no re-sequencing of the jobs between stations and all of the earliness and tardiness costs are constant. In effect, the lack of consideration of arrivals and non-permutation scheduling becomes a presupposition of FCFS. In this scenario Cheng concluded that an SPT sequence is optimal although this conclusion is at odds with the findings of Duenyas and Hopp below.

Duenyas and Hopp [18] proposed an analytical framework for evaluation of various job sequencing rules given that flow times can be optimally predicted. Working through a series of increasingly more generalized scenarios they concluded that an EDD sequence is optimal if the tardiness penalty is constant for all customers and proportional to the tardiness which seems to contradict Cheng [17] above. To achieve this result Duenyas and Hopp only assumed that pre-emption does not take place. The result of an EDD sequence being optimal is useful in that it provides direction for redesigning the workflow system in this author's construct to encourage EDD processing order but is not helpful in determining the optimal due-dates.

Similar to Duenyas and Hopp above, Lawrence [19] presupposed that the practitioner either has a simple system with closed-form flow time estimates, or has some way to determine flow times for complex systems. With that as a precondition, he described an analytical approach to setting due-dates based on previously observed forecasting errors. While Lawrence proposed to fit the forecasting errors, which he refers to as "G", using a Ramberg-Schmeiser distribution, he concludes that Erlang and Gaussian distributions worked equally well in his research. Lawrence made three observations that are particularly germane in this context: (1) exponential smoothing of the forecasting error distribution parameters enhances the accuracy of the fit, especially in time-dynamic situations, (2) various measures of performance lead to differing uses of the error distribution, e.g. Mean Absolute Lateness is minimized by adding the median of the error distribution to the predicted flow time, Mean Square Lateness is minimized by adding the mean of the distribution to the predicted flow time, and service level matching is met by adding the target percentile of the distribution to the predicted flow time, e.g. $G^{-1}(0.9)$ for a 90% Service Level, and (3) the analytic due date quoting policies that include information about the current system state outperform those that do not at least in the simple scenarios that the author evaluates specifically. Additionally, Lawrence's paper provided a good summary of the most common analytic quoting policies which will be useful for comparison with this author's proposed modeling-based approach.

van Ooijen and Bertrand [20] introduced a distinction in terminology intended to allow some leeway between the tightly estimated Internal Due Date (IDD) and the slightly looser

External Due Date (XDD). To set this difference, which is analogous to e_i in the problem description from Chapter Three, or the Safety Time from Alfieri, or Lawrence's error distribution, G, the authors proposed to adjust the XDD using the ratio of the current level of work in progress (acwip) to the average level of work in progress (nwip). Using variations of this quoting policy various sequencing rules were applied and the optimal cost per order was established over a variety of relative earliness/tardiness combinations. Van Ooijen and Bertrand's results brought some closure to the disagreement between Cheng [17] and Duenyas [18] by noting that when earliness and lateness penalties are of similar magnitude then SPT sequencing works best; however, when tardiness penalties are much larger than earliness costs a Due Date sequencing rule is best. Another interesting conclusion that can be drawn from the data is that in spite of the dependence on FCFS sequencing in much of the literature, FCFS provided among the worst performance of the sequencing rules tested.

Rajasekera, Murr, et al [21] opened by observing that including more information into the dynamic flow time prediction process produces better results. Much of the paper subsequently focused on an analytical description of a load-balancing algorithm that could be implemented in an information system integrated with the manufacturing system. The authors concluded that after applying their load balancing procedure and assuming FCFS processing, then setting due-dates is straightforward even when taking into account the jobs already in the system. As a parting note, the authors conceded that more complex work centers would require more complex queuing decomposition methods and further analysis.

		Literature Segment							Solution			Queuing	
Title	Primary Author	Business Process Modeling	Workflow Systems	Data Mining	Discrete Event Simulations	Embedded Simulations	Predictive use of Simulations	Due Date Quoting	Heuristic	Deterministic	Stochastic	First Come, First Served	General
Due date quoting and scheduling interaction in production lines	Alfieri, A.							X	X			X	
Optimal assignment of slack due-dates and sequencing of jobs with random processing times on a single machine	Cheng, T.C.E.							X		X		X	
Survey of scheduling research involving due date determination decisions	Cheng, T.C.E.							X					X
Quoting customer lead times	Duenyas, I.							X		X			X[1]
Production scheduling	Elion, S.							X					X
Estimating flowtimes and setting due-dates in complex production systems	Lawrence, S.R.							X			X		X
A due-date assignment model for a flow shop with application in a lightguide cable shop	Rajasekera, J.R.							X		X		X	
Economic due-date setting in job-shops based on routing and workload dependent flow time distribution functions	van Ooijen, H.P.G.							X		X		X	

Figure 56 - Due Date Quoting literature grid

Conclusion

What appears to be missing in the literature is using simulation to make predictions about the system when the job parameters and specific queuing behavior are unknown and the historical data that describes these factors is intractable to all but robust data mining techniques to describe. The preceding tables summarize the literature reviewed by the author and makes clear the gap described.

APPENDIX B: PRACTICAL CONSIDERATIONS

A note about balancing detailed data collection with human behavior -- the author has many years and many negative experiences developing and delivering “ideal” solutions to real users and achieving sub-par performance. Specifically, requiring the users of systems to enter data, or perform synthetic tasks for something other than the direct benefit of the user. In light of this, the author eschewed the notion of having users of this new system indicate all of the gruesome details of their processing of the documents and went simply with a single recorded step of when the user was finished with his portion of the processing. As a result, the transactional data that captures the trajectory of a document through the system must first be processed before it can be dealt with using classical modeling techniques.

Practical consideration #1 - Limit auditable steps to inter-personal boundaries – no process steps/status changes while the object is still in the possession of one actor. This technique, in combination with the transparency of reporting, makes the system self-regulating. Each actor is judged on the amount of time that objects spend in their care so it behooves them to complete their steps efficiently and flow the object to its next step so as to “stop the clock”. When actor #1 completes a step and flows the object to actor #2, actor #1 is asserting that his step is complete and ready for actor #2 to begin. However, because actor #2 is now responsible for the processing time of the object he is motivated to ensure that the previous step was, in fact, completed and if not, actor #2 can return the object for completion or rework to actor #1 placing the processing time onus back on actor #1. Because the actors are self interested (trying to avoid the baleful eye of the process owner) they are internally

motivated to flow the objects through the process as quickly and accurately as possible, but this is regulated by the downstream actors not wanting objects stacked in their queues that are not ready for processing.

Practical consideration #2 - Limit imposition of “extra” button clicking to maximize success – As mentioned in the introduction of this work, the author’s experience has lead him to build systems that minimize the requirement for users to perform synthetic, i.e. non-direct-value added, tasks. In light of this, the subject system simply records when the user finishes with his portion of the processing. Button presses that seek to capture information beyond what is minimally required will be “fudged” unless you can automate the capture of the event (selecting which object is being acted upon though this assumes serial processing, when in reality several objects are in play simultaneously).

Practical consideration #3 - Determine useful metrics to measure the process – avoid the pitfall of pulling numbers that are easy to capture but do not provide any real insight into the business process.

Practical consideration #4 - Selection of the data type for dates and times. The author has found it much easier to err on the side of selecting a higher precision data type and not making full use of the precision when not needed, than attempting to overcome the limitations of a less-precise data type – especially after the system has been in production for some time.

Practical consideration #5 – it is possible that a human-centric (a.k.a. smart) identifier may be applied to an object in addition to its system generated ID. If such an identifier is

required, the practitioner should carefully consider the mechanism by which it is generated, and that mechanisms relation to attributes of the associated objects, i.e. if one of the attributes of an object is changed -- does that change the identifier? Or alternatively, once defined should the identifier be static? In particular and if at all possible, such an identifier should NOT be used as a foreign key either within the workflow system or, even more importantly, across system boundaries.

Practical Consideration #6 – the actors performing the process will all have differing ideas about the best way to communicate their requirements. In the first case, the users will focus on the requirements of the routine execution of the process and fail to mention the steps necessary for extraordinary circumstances leading to unhandled cases during execution. Conversely, there will be users who will focus on all of the possible extreme cases, no matter how unlikely, and fail to adequately describe the normal process. Users in either of these camps are best interviewed in multiple sessions.

LIST OF REFERENCES

[1] S. B. Johnson, *The secret of Apollo: systems management in American and European space programs*. Baltimore: The Johns Hopkins University Press, 2002.

[2] H. E. McCurdy, *Faster, better, cheaper: low-cost innovation in the U.S. space program*. Baltimore: The Johns Hopkins University Press, 2001.

[3] J. Dallien, W. MacCaull, and A. Tien, "Initial work in the design and development of verifiable Workflow Management Systems and some applications to health care," Budapest, Hungary, 2008, pp. 78-91.

[4] M. W. Maier and E. Rechtin, *The Art of Systems Architecting*, 2nd ed. ed. Boca Raton: CRC Press LLC, 2000.

[5] A. Pourabdollah, T. Brailsford, and H. Ashman, "A user-oriented design for business workflow systems," Berlin, Germany, 2007, pp. 285-297.

[6] M. H. Larsen and R. Klischewski, "Process ownership challenges in IT-enabled transformation of interorganizational business processes," Big Island, HI., United states, 2004, pp. 4047-4057.

[7] H. Kilov, "From semantic to object-oriented data modeling," Morristown, NJ, USA, 1990, pp. 385-393.

[8] L. Yu, B. Chen, and J.-M. Xiao, "Multi-policy access control model for workflow management system," *Xitong Gongcheng Lilun yu Shijian/System Engineering Theory and Practice,* vol. 29, pp. 151-158, 2009.

[9] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come* Washington D.C.: IEEE Press, 2007.

[10] W. Mendenhall and T. Sincich, *Statistics for Engineering and the Sciences*, 4th ed. ed. Upper Saddle River: Prentice Hall, Inc., 1995.

[11] J. S. DeKeyrel, "Processing predictions through embedded simulation," in *Proceedings of the IASTED International Conference, Software Engineering and Applications (SEA 2010)*, Marina del Rey, CA, 2010, pp. 412-418.

[12] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. Alves de Medeiros, M. Song, and H. M. W. Verbeek, "Business process mining: An industrial application," *Information Systems,* vol. 32, pp. 713-732, 2007.

[13] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support," *Data and Knowledge Engineering,* vol. 68, pp. 834-850, 2009.

[14] T. C. E. Cheng and M. C. Gupta, "Survey of scheduling research involving due date determination decisions," *European Journal of Operational Research,* vol. 38, pp. 156-166, 1989.

[15] S. Eilon, "Production scheduling," in *Operational Research '78: Eighth IFORS International Conference on Operational Research*, North-Holland, Amsterdam, 1978, pp. 237-266.

[16] A. Alfieri, "Due date quoting and scheduling interaction in production lines," *International Journal of Computer Integrated Manufacturing,* vol. 20, pp. 579-587, 2007.

[17] T. C. E. Cheng, "Optimal assignment of slack due-dates and sequencing of jobs with random processing times on a single machine," *European Journal of Operational Research,* vol. 51, pp. 348-353, 1991.

[18] I. Duenyas and W. J. Hopp, "Quoting customer lead times," *Management Science,* vol. 41, pp. 43-43, 1995.

[19] S. R. Lawrence, "Estimating flowtimes and setting due-dates in complex production systems," *IIE Transactions (Institute of Industrial Engineers),* vol. 27, pp. 657-668, 1995.

[20] H. P. G. van Ooijen and J. W. M. Bertrand, "Economic due-date setting in job-shops based on routing and workload dependent flow time distribution functions," *International Journal of Production Economics,* vol. 74, pp. 261-268, 2001.

[21] J. R. Rajasekera, M. R. Murr, and K. C. So, "A due-date assignment model for a flow shop with application in a lightguide cable shop," *Journal of Manufacturing Systems,* vol. 10, pp. 1-7, 1991.

[22] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. ed.: McGraw-Hill, 2000.

[23] W. D. Kelton, R. P. Sadowski, and D. T.Sturrock, *Simulation with Arena*, 3rd ed. ed. New York: McGraw Hill, 2004.

[24] C. Azzaro-Pantel, L. Bernal-Haro, P. Baudet, S. Domenech, and L. Pibouleau, "A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm," *Computers and Chemical Engineering,* vol. 22, pp. 1461-1481, 1998.

[25] H. A. Reijers, van der Aalst, W.M.P., "Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making," in *IASTED International Conference - Modeling and Simulation (MS '99)*, Philadelphia, Pennsylvania - USA, 1999, pp. 417-421.

[26] K. Rojanapibul and J. Pichitlamken, "Assessing risk in a job schedule: Integrating a scheduling heuristic and a simulation model to a spreadsheet," Orlando, FL, United states, 2005, pp. 2136-2140.

[27] G. R. Cates and M. Mollaghasemi, "The project assessment by simulation technique," *EMJ - Engineering Management Journal,* vol. 19, pp. 3-10, 2007.

[28] J. DeKeyrel, C. Geiger, L. Malone, S. Lackey, and M. Mollaghasemi, "Real-Time Assignment of Due Dates within Workflow Management Systems," in *(pre-press) Industrial Engineering Research Conference*, Reno, NV, 2011.

[29] D. M. R. Ferreira and J. J. P. Ferreira, "Developing a reusable workflow engine," *Journal of Systems Architecture,* vol. 50, pp. 309-324, 2004.

[30] B. Milainovic, K. Fertalj, and I. Nizetic, "On some problems while writing an engine for flow control in workflow management software," Dubrovnik, Croatia, 2007, pp. 489-494.

[31] B. Gladwin and K. Tumay, "Modeling business processes with simulation tools," Buena Vista, FL, USA, 1994, pp. 114-121.

[32] J. F. Cook, J. W. Rozenblit, A. K. Chacko, R. Martinez, and H. L. Timboe, "Meta-Manager: A requirements analysis," *Journal of Digital Imaging,* vol. 12, pp. 186-188, 1999.

[33] J.-S. Bae, S.-C. Jeong, Y. Seo, Y. Kim, and S.-H. Kang, "Integration of workflow management and simulation," *Computers and Industrial Engineering,* vol. 37, pp. 203-206, 1999.

[34] A. Ghanmi, "Modeling and analysis of a Canadian Forces Geomatics division workflow," *European Journal of Operational Research,* vol. 170, pp. 1001-1016, 2006.

[35] Q. Jianhua, J. Zhibin, Z. Guotong, M. Rui, and S. Qiang, "A surgical management information system driven by workflow," Shanghai, China, 2006, pp. 1014-1018.

[36] H. Kayser, J. McIntosh, I. Williamson, and J. Hanson, "New generation well project management application improves cycle time, workflow efficiency, corporate compliance, and knowledge sharing (and people like it!)," Amsterdam, Netherlands, 2008, pp. 88-99.

[37] A. Abecker, A. Bernardi, H. Maus, M. Sintek, and C. Wenzel, "Information supply for business processes: Coupling workflow with document analysis and information retrieval," *Knowledge-Based Systems,* vol. 13, pp. 271-284, 2000.

[38] H. A. Reijers, M. Song, and B. Jeong, "Analysis of a collaborative workflow process with distributed actors," *Information Systems Frontiers,* vol. 11, pp. 307-322, 2009.

[39] G. Jiang and L. Dong, "Comparative research of modeling methods for workflow process," Guangzhou, China, 2008, pp. 976-980.

[40] G. Mentzas, C. Halaris, and S. Kavadias, "Modelling business processes with workflow systems: An evaluation of alternative approaches," *International Journal of Information Management,* vol. 21, pp. 123-135, 2001.

[41] V. Gruhn and M. Schneider, "Workflow management based on process model repositories," Kyoto, Jpn, 1998, pp. [d]379-388.

[42] C. Ames, S. Burleigh, S. Mitchell, and T. Huynh, "Applications of Web-based workflow," Big Island, HI, USA, 1998, pp. 79-87.

[43] H. Liang, Q. Wu, and J. Shi, "Research of Web-based workflow management system," *High Technology Letters,* vol. 7, pp. 55-58, 2001.

[44] L. Hong Va, H. Kei Shiu, and W. Lam, "Web-based workflow framework with CORBA," *Concurrent Engineering Research and Applications,* vol. 9, pp. 120-130, 2001.

[45] Y. Jin, Z. Wu, S. Deng, and Z. Yu, "Service-oriented workflow model," Taipei, Taiwan, 2005, pp. 484-488.

[46] D. Wan, Q. Li, and G. Chen, "Research and implementation of workflow interoperability crossing organizations," Nanjing, China, 2005, pp. 397-402.

[47] P. Muth, J. Weissenfels, M. Gillmann, and G. Weikum, "Integrating light-weight workflow management systems within existing business environments," *Proceedings - International Conference on Data Engineering,* pp. 286-293, 1999.

[48] C.-Y. Huang, "Distributed manufacturing execution systems: A workflow perspective," *Journal of Intelligent Manufacturing,* vol. 13, pp. 485-497, 2002.

[49] M. Sayal, F. Casati, U. Dayal, and M.-C. Shan, "Integrating workflow management systems with business-to-business interaction standards," San Jose, CA, United states, 2002, pp. 287-296.

[50] R. A. Botha and J. H. P. Eloff, "A framework for access control in workflow systems," *Information Management and Computer Security,* vol. 9, pp. 126-133, 2001.

[51] L. Lin, Y.-Z. Zhan, and Y. Nian, "Improved RBAC model based on organization chart," *Jiangsu Daxue Xuebao (Ziran Kexue Ban) / Journal of Jiangsu University (Natural Science Edition),* vol. 27, pp. 147-150, 2006.

[52] L. Chen and D. Feng, "Study of information security in workflow management system," *Yi Qi Yi Biao Xue Bao/Chinese Journal of Scientific Instrument,* vol. 28, pp. 432-436, 2007.

[53] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Boca Raton: Chapman & Hall/CRC, 1984.

[54] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems,* vol. 2, pp. 303-314, 1989.

[55] V. N. Vapnik, "Support vector method," in *Proceedings of the 1997 7th International Conference on Artificial Neural Networks, ICANN'97, Oct 8 - 10 1997*. vol. 1327 Lausanne, Switzerland, 1997, pp. 263-263.

[56] K. P. Bennet and C. Campbell, "Support Vector Machines: Hype or Hallelujah?," *SIGKDD Explorations,* vol. 2, pp. 1-13, 2000.

[57] W. Chen, C. Ma, and L. Ma, "Mining the customer credit using hybrid support vector machine technique," *Expert Systems with Applications,* vol. 36, pp. 7611-7616, 2009.

[58] C.-C. Chiu, C.-C. Tien, and Y.-C. Chou, "Construction of clustering and classification models by integrating Fuzzy ART, CART and neural network approaches," *Journal of the Chinese Institute of Industrial Engineers,* vol. 22, pp. 171-188, 2005.

[59] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing,* vol. 55, pp. 169-186, 2003.

[60] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, 2009.

[61] A. Shanker and W. D. Kelton, "Empirical input distributions: an alternative to standard input distributions in simulation modeling," in *Simulation Conference, 1991. Proceedings., Winter*, 1991, pp. 978-985.

[62] E. M. Jewkes and A. S. Alfa, "Empirical discrete distributions in queueing models," Marina Del Rey, CA, United states, 2004, pp. 42-45.

[63] J. M. Garrido, *Object-oriented discrete-event simulation with Java: a practical introduction*. New York: Kluwer Academic/Plenum Publishers, 2001.