ANALYSIS OF A MINE-MILL PRODUCTION SYSTEM USING SIMULATION AND INTEGER PROGRAMMING

by

Jun Zhou

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
November 2010

Dalhousie University

Faculty of Engineering

Department of Industrial Engineering

The undersigned hereby certify that they have examined, and recommend to the Faculty of Graduate Studies for acceptance, the thesis entitled "**Analysis of a Mine-Mill Production System Using Simulation and Integer Programming**" by **Jun Zhou** in partial fulfillment of the requirements for the degree of **Master of Applied Science**.

Dated: November 30, 2010

Supervisor: _____
Dr. Eldon Gunn

Co-supervisor: _____
Dr. John Blake

Examiners: _____
Dr. Don Johns

_____

Dalhousie University

Faculty of Engineering

DATE: November 30, 2010

AUTHOR: Jun Zhou

TITLE: Analysis of a Mine-Mill Production System Using Simulation and Integer Programming

DEPARTMENT OR SCHOOL: Department of Industrial Engineering

DEGREE: M.A.ScCONVOCATION: May YEAR: 2011

Signature of Author

Table of Contents

List of Tables

List of Figures

Abstract

Mine-mill production faces several operational difficulties, such as fluctuations in ore delivery from mines, random failure of machines, usage of stockpiles and storage bins, and changeover time when switching products. This study was initiated at a particular Canadian mining company. However, changes in the economic condition in the mining industry during 2008 have meant that circumstances have changed to the extent that this work should be seen as an illustration of methods rather than a study of the specific situation at the mining company. This mining company will remain unnamed throughout this thesis. The purpose of this research is to develop a series of production campaigns, each of which uses a specific draw scheme to coordinate the receiving of ore, maintenance planning and product scheduling.

The approach includes a combination of mathematical programming model and a simulation model. The solution from the integer programming model is a set of campaigns that minimize the inventory levels of unprocessed ore, the number of days on shutdown, and the number of active piles required at any point in time. The simulation model uses this solution as its production scheduling input with integrated stochastic elements to evaluate mill system performance.

In this thesis, the formulation of the mathematical programming model and construction of the simulation model, as well as the maintenance data analysis used as stochastic element of the model is discussed.

List of Abbreviations Used

DES	Discrete Event Simulation
FOB	Fine Ore Bin
GLPK	GNU Linear Programming Kit
IP	Integer Programming
LHD	Load Haul Dump
MIP	Mixed-Integer Programming
MTBF	Mean Time Between Failure
MTTR	Mean Time to Repair
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
ROM	Run of Mine
VBA	Visual Basic Application

Acknowledgements

First, I thank my co-supervisor Dr. Eldon Gunn for his continuous support in the Master of Applied Science program. He helped me to understand the overall control framework and campaign draw scheme concepts. He showed me the formulation of integer programming model and helped me to implement the model using GUSEK and the GNU Linear Programming Kit. Dr. Gunn was always there to listen and to give advice. Without him, this thesis would not have been possible.

A special thanks goes to my co-supervisor, Dr. John Blake, who is most responsible for helping me implement the simulation model in Visual Basic as well as complete the output analysis. He introduced the station concept used in the simulation model and made me a better programmer. Without his encouragement and constant guidance, I could not have finished this thesis.

Dr.Don Jones from Mining Department, Dalhousie University provided me with valuable feedback about my thesis. I appreciated his help in suggesting changes that made this thesis more readable.

Besides my supervisors, I would like to thank several personnel from the mining company: Mr.Peter R, who supervised my work at the Mill and evaluated the simulation model and gave helpful feedback, Mr. Chris L and Ms. Marta B, who advised on mill operation policies.

I am thankful to Dr. Claver Diallo , for helping me analyze the maintenance data during the research. Let me also say thanks to the following people at Industrial Engineering Department of Dalhousie University: Cindi Slaunwhite and Mary-Anne Wensley for their kind support.

I would like to show my gratitude to the work done by GNU group for helping the development of free software, and particularly GLPK Project. And thanks to the GUSEK project for providing an open source LP/MILP IDE for Win32, packing a custom version of the SciTE editor linked to the GLPK standalone solver.

Last, but not least, I thank my parents: Xuenian Zhou, and Zhengyi Lu, for their

supporting me during my studies in Halifax, Canada and for unconditional understanding and encouragement to pursue my interests in the field of engineering.

Chapter 1

Introduction

1.1 A Canadian Mining Company

This research project was conducted at a Canadian mining company. Nickel and copper are the primary metals, but cobalt and precious metals such as platinum are also produced by the company. In 2008, the market for nickel changed dramatically. The mining company closed several mines in the study area and reduced its operation to nickel copper mines, a mill complex and a smelter.

1.2 Mill Complex Operation

1.2.1 Overview of Mill Production System

The mill complex receives ore from mines and produces two concentrate streams: a copper concentrate and nickel concentrate, which are transported to a smelter for processing into nickel metal and copper metal. Currently, the basic flow of material starts from the mines, where ore, crushed (depending on the mine), is trucked to the mill site. The ore is then either stockpiled in several stockpile or fed directly into one of the three feeders at the mill. The feeders are the entry points for the ore into the mill. The ore is carried by conveyor to the crushing process where it is initially screened. Particles fine enough to bypass the crushing process go directly onto the grinding process, while, large pieces of ore are crushed by the initial crusher and screened through two screens. If the ore is still oversize, it gets sent to one of two secondary crushers and continues in this cycle until the particles are fine enough to pass through the screen and enter the grinding process. Once the ore is sufficiently fine, it is conveyed to the fine ore bins and stored in one of five ore bins. The three

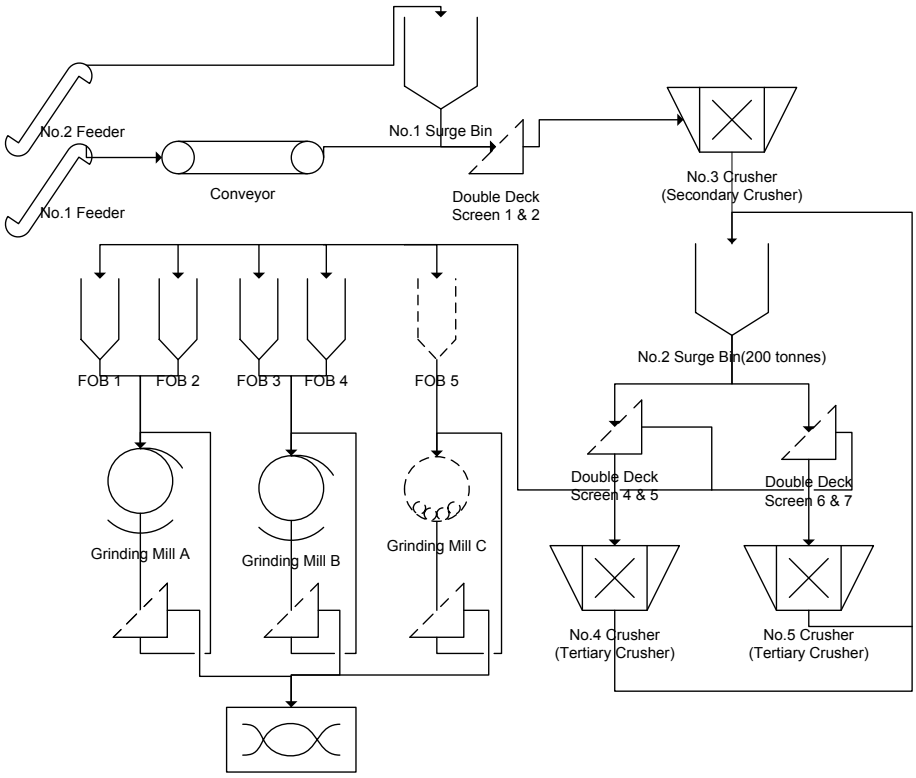

Figure 1.1: Mill Process Map

grinding circuits (A, B and C) have the flexibility of processing fine ore from any of the fine ore bins. Currently, in order to reduce the planning complexity, only two circuits (A and B) and four fine ore bins (FOB 1-4) are used. Moreover, circuit configurations specified for FOB 1 and 2 feed into circuit A while FOB 3 and 4 feed into circuit B under current operating policy. The grinding circuit consists of a rod mill, in which the fine ore is ground to a finer size and mixed with water to form slurry. If it does not meet the size requirements, it is sent to a ball mill and is screened again. The ball mill process is repeated until the material meets the size requirements for the floatation process. The floatation process separates minerals by taking advantage of differences in their hydrophobicity prior to refinery process. The following section explains in detail the different process units within the mill. The mill process map is presented in Figure 1.1:

Mine Product

There are three different types of "products" being processed in the mill complex: copper ores, nickel ores and custom feed. Copper and nickel ores comes from the mines operated by the mining company. Custom feed includes ore from other companies' mines. Custom feed can be broken into various subproducts. One product cannot be mixed with another; the entire circuit, from the feeder to the fine ore bin, has to be emptied whenever it processes a different type of product.

Stockpile

Stockpiles are used to store crushed or un-crushed ore. The stockpiles are all located on the mill site property and are in close proximity. One concern is to minimize the number of stockpiles being used. Less number of active piles can speed up the movement of the ore and affect the "shelf life" of ore. Shelf life refers to the maximum time ore can be exposed to air before significant deterioration of the sulphide in the ore due to oxidation.

Feeder

There are two feeders feeding material into the mill. The No. 1 feeder is the main entry point of material, with a 400-tonnes-per-hour transfer rate. The No. 2 feeder feeds only from the No. 2 stockpile by loader.

Crushing

The mill circuit starts with the secondary crushing process. All material, either from the No. 2 feeder or the No. 1 feeder, goes through the secondary crusher (No. 3 Crusher). The surge bin (No. 1 Surge bin) associated with the secondary crusher has a capacity of 200 tonnes. Before material enters the secondary crusher, it is screened through a double-deck screen (Screens 1 and 2). Ore that is fine enough to pass the screens is allowed to bypass the crushing process and go directly to the fine ore bin. The rest of the ore on the screens goes through the secondary crusher.

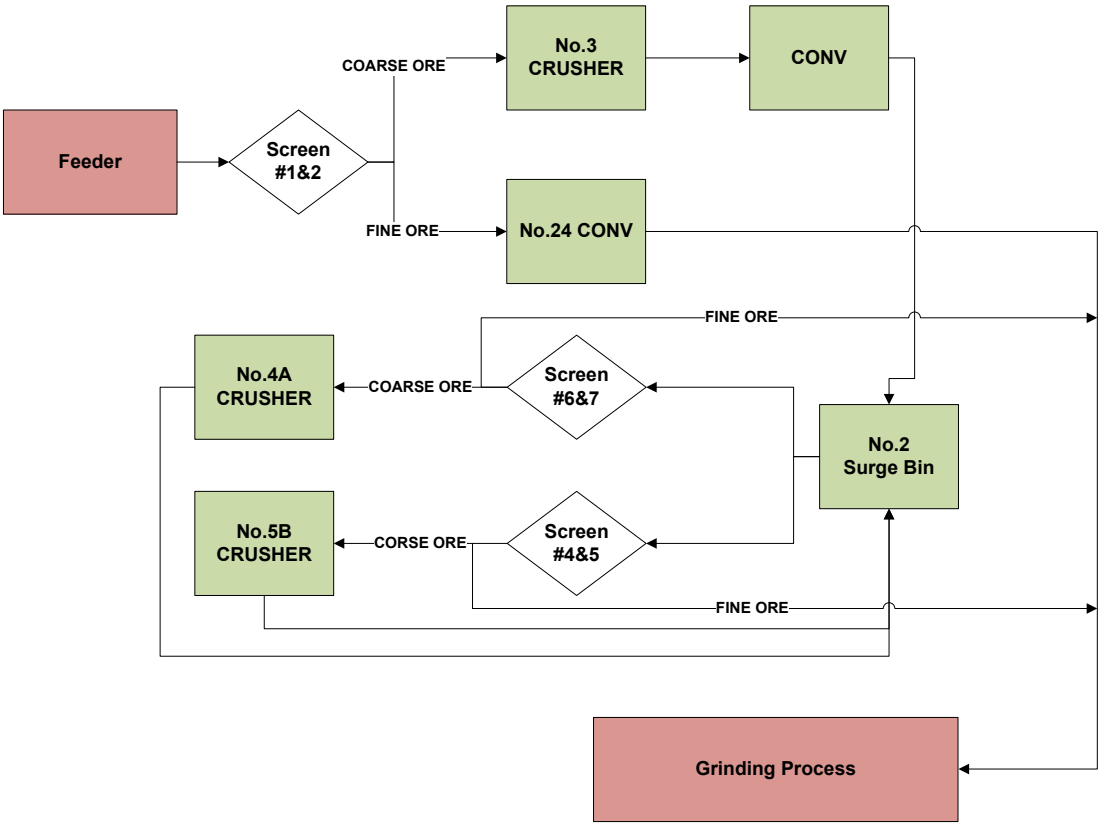

Figure 1.2: Mill Crushing Process Flow Block Diagram

All material, after being crushed in the secondary crusher, get conveyed by conveyor to another surge bin (No. 2 Surge Bin) which has a capacity of 200 tonnes. This surge bin feeds material to two sets of screens (Screens 4 & 5 and Screens 6 & 7). Material that is fine enough to pass through these screens is conveyed to the fine ore bin. Otherwise, the ore is sent to the tertiary crusher (No. 4A and No. 5B crushers) for further crushing. All material, after passing through the tertiary crusher, is sent back to the No. 2 Surge Bin and Screens 4 & 5 and Screens 6 & 7 again. This process is repeated until the material is fine enough to pass through the screens. The crushing process is presented in the process block diagram (Figure 1.2). Physically, the input to the crushing process is ore received from either mines or customer feed. The outputs are crushed ore that meet the required size of grinding process. There are two fundamental activities in crushing process: screening and crushing. The crushing process loops between these two activities until ore is fine enough. The looping process is implemented by applying an essential control policy

for crushing operation: once ore enters a crushing process, either at surge bin, screen or crusher, only the type of ore can be processed until it leaves the system entirely. In other words, crushing process can only process one type of ore at a time.

From Figure 1.2, the looping process starts from the first screen set (1& 2), and ends at the conveyors before grinding circuits. The decision that need be made is which and how much material should be fed into the crushing process and what throughput can be expected from this process. As a consequence, reduced throughput is expected from crushing loop process. In the later chapter and simulation model, instead of focusing on analyzing the complexity of crushing process, the crushing process is aggregated into one production unit.

Fine Ore Bins

There are five fine ore bins in the mill, four of which are in use. Each bin has a capacity of between 2,500 and 2,800 tonnes, for an overall storage capacity of 10,000 tonnes. Each bin can contain only one product at any time. Therefore, bins must be completely emptied before they can be filled with a different type of product. One conveyor moves back and forth to fill all the bins. Each bin must be completely filled with product before another bin is started.

Grinding Circuit

While there are three grinding circuits in the mill, only A and B are presently in use. Circuits A and B are basically identical, with a throughput of 400 tonnes per hour each. Grinding circuits A and B consist of a rod mill where the fine ore is made into a slurry-type mixture. The slurry is then screened. If it does not meet floatation requirements, the material is cycled through the filter again and goes on to the rod mill until it is broken down sufficiently enough.

The future configuration of the grinding circuit will include the fifth fine ore bin and grinding circuit C. The fifth fine ore bin will feed grinding circuit C. Ordinarily, bins 1 and 2 feed grinder A only, and bins 3 and 4 feed grinder B only. Grinding circuit C will consist of a ball mill only. Material leaving the mill will be screened, and

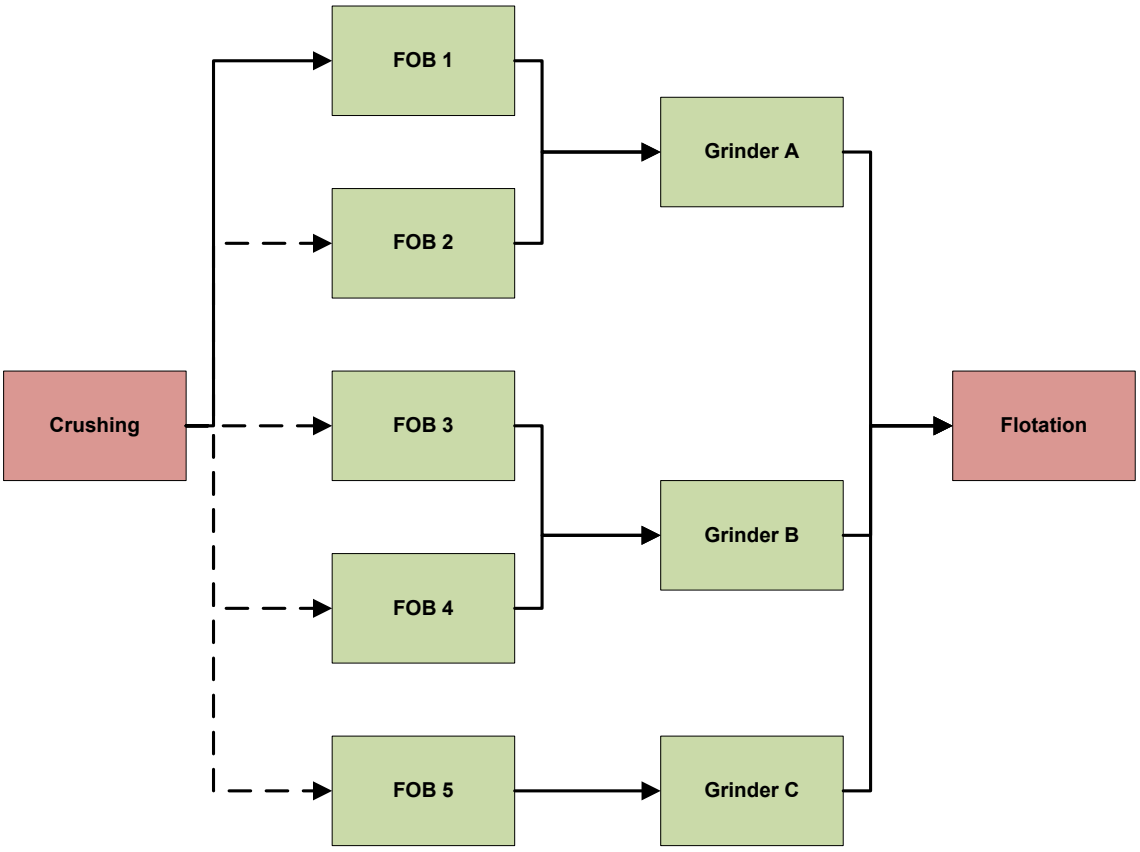

Figure 1.3: Mill Grinding Process Flow Block Diagram

oversize material will be sent back through the C ball mill again. This process will be repeated until all material is appropriate for flotation. In general, each FOB in the grinding process has the capability to feed material into any grinder. Planning in the mill currently uses the configuration in Figure 1.3 to reduce the level of complexity. Under this configuration, it is preferable to keep the same product in both fine ore bins 1 and 2. This also applies for bins 3 and 4. If need be, bins 1 and 2 can contain different products, but one must completely empty and go through the grinding circuit before the ore in other bin is moved.

1.2.2 Mill Production Planning Problem

Mill production planning is very important. "Planning" means "Predictability", since it is necessary to verify if a plan can be executed or not, is in agreement with the available resources, given fluctuations in receiving, and has the largest possible throughput.

Below are listed some issues and challenges faced by the mill in its operation planning.

Oxidized Material

The "shelf life" relates to the oxidation of ore as a result of exposure to oxygen in the air. If the ore is exposed to air for extended periods of time, the oxidized ore in the slurry (during the floatation process) will have a lower PH value which will negatively affect the mineral recovery, due to change in chemistry of mineral surface. As a result, lime must be added into the floatation circuit to increase PH to maintain selectivity of the valuable minerals. This increases lime consumption and can create environmental issues for the mill. Currently, the ore can be exposed to air for two weeks without significant oxidation.

The importance of limiting oxidation is related to mill production scheduling. The key point is to ensure that the ore received is processed in a timely manner. Since the mill operates with a range of mill runs(campaigns), feed for the different campaigns are stored separately. The issue of delay in processing ore results in a backlog of ore being stored in the stockpile area for longer periods of time, resulting in increased oxidation. The target of mill production scheduling is to process all of the material that is available in the stockpiling area. However, it is difficult to achieve this target due to unpredictable machine failures in the mill process. If one type of product is not used up entirely and the next campaign, with a different product, goes ahead as scheduled, a backlog of ore in the stockpiling area is created. More importantly, the newly-arrived fresh ore is often stored in front of a previous pile, thus blocking access to bring the older ore piles to the mill. As of January 2007, the mill had approximately 10,000 tonnes of oxidized ore that had been stored in the stockpiling area since June 2006. It is important to have production scheduling and campaign schedules set up so that the fluctuations affecting the circuit are minimized.

Blending Option

Currently, the mill complex has two options to blend ore: the stockpile area and the fine ore bin. At the stockpile area, ore that is piled on the stack will be the first

taken out to be fed to the mill. As the truckloads of ore from the different mines arrive, they are dumped in the stockpile area in discrete rows depending on the type of ore. A second row is created in front of the first row. A loader moves the second row towards the first one so that blending takes place. A third row is thus created on the opposite side of the first row.

Within the grinding circuit, the target for blending in the rod mill is where the ratio of Cu and Ni content is 1:1. The mill has an automated system where the speed of the conveyor is adjusted so that the feed rates of the copper and nickel ores with different grades going into the rod mill are adjusted to meet the target 1:1 Cu:Ni ratio.

The blending option is another important aspect for mill production. The stockpile blending option has more flexibility than fine ore bin blending since it is done on the surface and requires only a loader to finish the blending. The mixed ore can then be fed into the mill without worrying about fine ore bin allocation. The entire circuit only has one type of product in this case. Fine ore bin blending offers the controllability of which type of ore feed to which grinding circuit. The biggest challenge in using this blending strategy is finding the best way to empty the fine ore bins when switching between different types of feed.

Partial Failure

An interesting aspect of mill operation is that when a machine breaks down, the throughput of the machine is typically not zero. Often it runs at partial throughput until it is taken off-line and repaired. The throughput levels during this period will be less than 100% and vary based on the nature of the machine failure. When the failure efficiency is zero, the machine is completely down, with zero throughput. Any other number indicates that the machine is running at a percentage of full availability. The reason to emphasize failure efficiency is its impact on the transfer rate. When a machine works without a failure, the transfer rate is 100% of its regular transfer rate; otherwise, the transfer rate is reduced. The detailed discussion of partial failure is discussed in Chapter 5.

1.3 Production Scheduling - Draw Scheme and Campaign

The mill production process has many issues and challenges, some of which are discussed above. Some of these are stochastic, some are deterministic. Some of the issues that impact on production include:

i) Received material from different mining areas has different ore types and grades.

ii) The need to extract correct quantity of material from each stockpile to reach the desired quality (grades) and quantity.

iii) The number of types of ore in the circuits is limited by the configuration of the circuit and the feed.

iv) The fine ore bins and grinding mill limit grinding circuit's flexibility.

v) Random failures of equipment.

vi) Maintenance schedules.

There are several essential levels of production control exerted within the mill. These include:

i) Control of the overall global ore type being fed through the surge bin and crusher.

ii) Control of the overall draw of material due to the choice of how the grinders are being fed from the fine ore bins.

iii) Control of which of the fine ore bins (FOBs) to direct the current feed to.

iv) Maintenance schedules and policies.

The solution to mill production planning in this research introduces the concept of a campaign and its draw scheme. A draw scheme consists of specifying, for each grinder, either an ore of a given type or a blend of a given ratio from two or more

fine ore bins. The campaign is defined as a consecutive number of days for which we use the same draw scheme. We expect that the overall planning horizon will contain several potential campaigns, and that each campaign uses one draw scheme only during the entire campaign period. This situation can be modelled as an optimization module, focused at defining the length of campaigns and the draw scheme used for each campaign, while estimating inventory level of stockpiles.

A discrete-event simulation model is used to reproduce the randomness of the equipment availability. This makes simulation a great tool to check the feasibility of the mill production schedules. But the simulation tool itself is not able to precisely select the decision logic used to choose the best campaign and related draw scheme. To develop the campaign concept requires an optimization model. The proposed solution is the integration of both tools: discrete-event simulation and optimization.

1.4 Project Background

When the research project was originally started, the initial intention was to solve the issue of operating the mill at a very high level of throughput. The target was to process approximately 3.4 million tonnes of ore per year through the mill. The main focus of the research project was to improve the utilization of the grinding circuit to ensure that the mill had the ability to process the larger amount of material it received.

A significant change happened about 8 months later. As a result of the world-wide drop in price of nickel, there were only two mines in production. The mill revised plans to process less than 1.5 million tonnes annually. The emphasis for the project changed from higher utilization of the grinding circuit to scheduling production campaigns and interspersing these with shutdown periods to improve production efficiency.

The purpose of this research project was defined as developing an integrated approach to combine simulation modelling with scheduled optimization to analyze the throughput of the mill complex. The research was to focus on the development of a suite of tools that would permit the company to model and control the dynamics of

ore delivery, sequence and mill operation and thus to make better operational decisions, which would result in increased yields and greater margins. The goal of this research was to:

i)Conduct a literature review on mining simulation, mill production planning and mine production optimization.

ii)Construct an integer programming model to calculate draw schemes and campaign schedules for the comminution processes in the mill.

iii)Provide a simulation model to evaluate a given scheme of controls at the mill. The model was to provide insight to coordinate ore delivery, maintenance planning, and product schedules.

iv)Test the simulation model and report on results.

The original outcome of the project was to develop a specific application for this specific mining company. Due to the significant changes at the company, the people who originally started the project are no longer involved. This changed the research project to the extent that this work should now be seen as an illustration of methods, rather than a study of the specific situation for a specific mining company. Hopefully, this project can be of interest to others and used as a starting point in their analysis of mill production systems.

Based on confidentiality agreements with the company, the data used in the research are made up for analytical purpose. Readers should be aware that number presented in this thesis do not represent the data that originated at the company. In addition, readers should not assume that the physical layout or the situations described actually represent either the existing or previous situation of the company. The company, while we were working with it, was considering various alternative designs for the mill complex. The system represented here is meant to be a realistic possible configuration but may not necessarily represent the actual mill or any of the planned modifications.

The next chapter presents a literature review on topics of mining simulation and mining production scheduling. The remainder of this thesis is organized as follows. In

Chapter 3, formulation of integer programming model is presented. The development of the simulation model is presented in Chapter 4. Chapter 5 presents simulation input data analysis of machines failure. Chapter 6 presents a simulation-based sensitivity analysis. Chapter 7 draws conclusion for this research and propose a direction of future study.

Chapter 2

Literature Review

In this chapter,a discussion of the history, current status and future development of the application of simulation in the mineral industry is given. Several case studies completed in Europe, South Africa and Canada are reviewed. The second section discusses in detail two simulation case studies completed at the Ervei Khuder Mine and the Vale Aguas Claras Mine to demonstrate how a simulation model is applied for production scheduling and planning purposes. The third section reviews the modelling and optimization of a crushing plant. A review of evolutionary algorithms used to optimize a crushing plant's performance with specific objectives and constraints is provided. Finally, the fourth section provides the review of a paper describing the determination of a production schedule at the LKAB Kiruna Mine by mixed-integer programming.

2.1 Simulation in the Mining Industry

Simulation has gained increased attention as a technique for planning and analyzing mining operations, or modifying and improving existing ones. Simulation can study the behaviour of a mining system before it is implemented. It evaluates design alternatives, identifies improvements, eliminates problems and justifies costs. Sturgul and Li [19] emphasized that the advantages of simulation in the mining industry are not only to provide management with a detailed look into the future but also to allow the company to make critical decisions and understand a variety of issues about the system.

2.1.1 The History of Simulation

Simulation in the mining industry has a close relationship with the development of the personal computer. Lynch and Morrison [13] reviewed the history of simulation in the mineral process along with milestones in PC development history. Early modelling (prior to 1960) is concerned with the design and optimization of circuits. Models focused on minimizing costs per tonne through increasing mineral recovery or productivity. However, little simulation progress was made at the time on the formulation of models, due to the lack of powerful computers to perform intensive and repetitive calculations. From 1960 to 1980, simulation studies focused on process analysis and control, in particular ball mill-hydro cyclone circuits. Since 1962, funding has increased and simulation projects have expanded to include all mineral processes. From 1980 to 2000, the combination of lower grades, rising mineral demands, and the requirement of high-capacity circuits to reduce unit costs led to the development of circuits with high-capacity, as well as rapid growth in machine sizes and power consumption. During this time, computers were already being used in plants, enabling simulation programs to be tested thoroughly. Because of this situation, many companies improved their working efficiency by using simulation models to improve process productivity and verify circuit design. According to Sturgul and Li [19], "the development and improvement of simulation tools increased interest in the mineral industry for simulation models." The authors listed several widely-used simulation tools employed in the mining industry:

i) **GPSS/H and Proof Animation**: the most widely-used classical general purpose simulation language.

ii) **SIMAN/Cinema/Arena**: the first major simulation language to be available for microcomputers which has both discrete and continuous modeling capabilities. Arena adds a graphical interface to SIMAN simulation.

iii) **SLAM/SLAMSYSTEM/AweSim**: SLAMSYSTEM is the first simulation language with graphical model building capability supported by the Windows platform.

iv) **MODSIM III**: An object-oriented, general purpose, high level programming simulation language.

v) **WITNESS**: has been used to develop an underground hard rock mining model. It has the ability to import 3D graphics files.

2.1.2 Current Status of Mining Simulation and New Development

Lynch and Morrison [13] suggested that there are two main reasons for simulation modelling successes: the availability of PCs since about 1985 and the comprehensive process data that had been collected in mineral processing plants over this time period.

On the other hand, Sturgul and Li [19] reviewed the most recent developments in simulation technology, listing four major advances in simulation technologies:

i) "Visualization and animation: the recent development in graphic modelling avoids complex programming in animation. Animation provides the ability to communicate the essence of a simulation model or the simulation itself to managers and other key project personnel, increasing the model's credibility. In addition, the use of visualization provides a more user-friendly interface and facilitates the process of simulation modelling for aspects such as debugging.

ii) Model reusability: differing from programming languages and general-purpose simulation languages, object-oriented simulation tools developed in recent years increase a module's reusability. This feature allows a developer-level user to construct the model, an analyst-level user to extend and reuse it, and an end level user to do experiments.

iii) Parallel simulation: for a large, complex simulation, parallel simulation is used for many large discrete event simulations such as communication systems, traffic systems, computer networks, and computer systems.

iv) Application integration: the development of Open Database Connectivity (ODBC) and Object Linking and Embedding (OLE)technology offers the ability to use simulation with other planning applications such as spreadsheets, CAD systems, databases, etc."

Lynch and Morrison [13] suggested that future research areas in simulation modelling in the mineral industry could include flotation, fine grinding and dry grinding. These authors concluded that simulation of flotation system has made progress on the prediction of concentrate grades but requires enormous and costly data collection surveys. The requirement for modelling dry grinding circuits is the same as for many other types of circuit. Fine grinding is coming into common use for re-grinding concentrates and improving recovery by leaching from refractory gold minerals. The authors also mentioned that economic factors have become more important in recent simulation modelling. Consideration for the cost of items such as equipment, power, labor, suppliers, and of the contract conditions for sales of products are used in simulation to improve economic performance. Lynch and Morrison [13] highlighted the following trends in simulation use in mining:

i) More flexible and user-friendly simulation languages that provide on-screen model building.

ii) Web-based simulation models.

iii) Integration of mining simulation and animation with mine design and planning applications.

In conclusion, mining simulation has been and will be improved in parallel with the rapid development of computer technology. With more powerful, flexible and user-friendly software development, simulation is a useful tool for system design, planning, and operations analysis in the mineral industry.

2.1.3 Mining Simulations Around the World

Mining Simulations in Europe

According to Panagiotou [17], European engineers have been pioneers in mining simulation since the late 1950s. The dramatic reduction in large-scale mine development in recent years, due to environmental issues and lack of profitable deposits, has had an impact on the application of simulation studies in Europe. Panagiotou [17] investigated several completed mine simulation studies in Europe. The early simulation studies in Europe focused on underground haulage systems. The first mine simulation is the transportation system for LKAB's Kiruna underground iron mine. The model consists of a trackway plan, ore storage bin, the signal system, train movement and dispatching. The simulation was done by hand in the late 1950s. In 1970, Wilke [25] developed a model for train transportation in underground coal mines in Germany. The objective was to optimize the underground traffic from three mines to one shaft. A simulation package called SIGUT was developed for this study. It is capable of handling the stochastic nature of mining data. The program was concerned with the material flow of both the belt system and truck haulage. The early simulation was further extended and Panagiotou [17] listed as follows:

i)Panagiotou developed a series of FORTRAN programs to simulate opencast mines operating with conveyors and stackers.

ii)Agioutantis [1] developed a simulator to study the performance of surface-mining equipment, which includes production subsystems, waste material, conveyor subsystems and dumping systems.

iii)Wilke [26] developed a program to minimize truck haulage cost which involves determining the optimum fleet haul size, organization of haul system, the best dispatching scheme and reliable maintenance and servicing system.

iv)Vagenas [22] developed METAFORA, a simulator for dispatch control in surface mines.

v)Panagiotou [16] presented a suite of programs to plan and analyze truck-/shovel operations in opencast mines and quarries.

vi)Medved [14] developed a simulation model in GPSS to study the truck transportation system which consists of traffic patterns, truck utilization operation costs,etc.

vii)Erdem [3] developed simulation models as part of an expert system for dragline and stripping method selection in surface coal mines.

viii)Mutagwaba [15] developed a model, written in C++, for simulating mine transportation systems.

In conclusion, simulation models for studying complex mining system is well accepted in Europe. The dynamic and stochastic nature of any mining system frequently makes simulation the only practical method for studying such systems.

Mining Simulations in South Africa

Turner [21] described a selection of simulation case studies and approaches in the mining fields in South Africa. The first example uses simulation as an aide to an on-line mining environment. South Africa is using a real time transport tracking and scheduling system in its mining industry. Simulation provides valuable input during various phases of an operational mine management system's life cycle. Turner [21] summarized that general simulation model purposes as followed:

i)Pre-implementation analysis: based on operational rules, the model is built and then verified to perform a what-if analysis for the decision-maker. The simulation results address improvements in availability and utilization as well as efficiency.

ii)Benchmarking of existing operations: the simulation model is used to establish a benchmark for measuring and quantifying operational improvements.

iii)Verification of assignments and operational logic: the simulation ensures optimal operational dispatching within constraints.

iv)What-if analysis: in a virtual mine simulation environment, what-if analysis assesses the impact of an altered pit or equipment configuration and/or operational rules.

Turner [21] also discussed several simulations of planning and operations in mineral processing in South Africa mining operations. The author lists numerous simulation models to evaluate operational alternatives:

i)De Beers Finsch Diamond Mine Quadrant Mining Simulation: implemented scheduled maintenance for both production and ground handling system. The simulation model evaluated the feasibility of operational options by obtaining the most suitable sequence in which the quadrant should be taken out of production.

ii)De Beers Block-Cave Mining Simulation: simulation of underground activities for a planned new block-cave development. Simulation compared the time and efficiency between two mining methods: drill-and-blast mining and high-speed continuous mining.

iii)Kimberly Mines Dump Relocation Simulation: simulation evaluated different logistic options and measured their consequence on the transport from mine dumps to the proposed new treatment plant.

Turner [21] discussed another simulation case study completed by JCI Gold and Uranium Division. The simulation model integrated traditional mine planning systems, mining and transport simulations, and financial costing systems. The model consisted of four sub-models:

i)Mining model: included all mining rules and was the heart of the entire simulation. Its output included a plan of how, where and when the ore body is to be mined.

ii)Financial model: associated various costs such as labour, stores, utility and others to each mining activity, with results being accumulated over time. Other business parameters such as revenue, inflation, interest, and gold price were also included.

iii)Capacity model: was used to report utilization of various ore transport routes to transport ore mined underground to the surface.

iv)Logistics model: a simulation model of the movement of people, material and ore between levels and shafts.

Turner [21] also mentioned that SASOL Limited, one of the world's largest coal mining operations, used simulation in their design and implementation of coal transport, storage and blending systems. This simulation focused on ensuring a long-term supply of coal from various mines to its Secunda Synthetic fuel operations. The model was used to evaluate marketing of coal production as export steam coal versus conversion to synthetic fuel operation. The option was to expand one of the existing mines. The objective was to minimize the disruption of other current mining operations. The simulation model evaluated the design of new mining, transport and handling systems for this expanded mine.

ISCOR Limited, a mining company producing iron ore, coal and heavy minerals, has used simulation extensively for mining and process investigations. Iron ore crushing and blending plants at the Sishen mine used simulation for planning. The blending system was described as one primary crusher at one end of the process and a feed to the process plant at the other. In between were a number of screening sections and two blenders. The simulation studied stockpile operation policies to determine best practices for the blending, stacking and material reclaim operations.

Turner [21] summarized that simulation has and is being used extensively in mining industry in South Africa. Increasingly, simulation incorporates the use of economic models into traditional dynamic models to assess the economic impact of operations. Overall, simulation in mining in South Africa is a growth industry.

Mining Simulation in Canada

Vagenas [23] explained that, through the 90's, the Canadian mining industry demonstrated increased awareness and interest in the application of discrete-event simulation software packages for both open-pit and underground mining operation. In Canadian open-pit operations, simulation modelling was primarily focused on analyses of materials handling systems. This application was related to truck/shovel dispatch systems and real-time management information systems. In underground mining, simulation studies have mostly focused on the equipment systems and the extraction rates of entire ore bodies or a section of a mine for development and production scheduling.

Vagenas [23] described trends in the use of simulation in the Canadian mining industry as follows:

i) Use of three-dimensional animation to visualize entire ore bodies - 3-D animation/simulation package such as AutoMod and Witness were preferred by mining engineers and consulting firms in Canada.

ii) Integration of reliability assessment analyses of mining equipment systems in simulation models.

iii) Integration of simulation with real-time mine management systems and spatial databases.

iv) Development of both strategic and tactical mine simulation models to provide insight into long and short-term requirements of mine operations.

Vagenas [23] concluded that simulation in the mining industry provides the ability to achieve "Lean Mining", which aims to minimize throughput time, stockpiles, wastage and rework. Three-dimensional animation was predicted by Vagenas [23] to be a dominant operational research tool for evaluation and comparison of current and future mining systems.

2.2 Application of Mining Simulation

2.2.1 Using Operation Process Simulation for a Six Sigma Project of Mining and Iron Production Factory

Chinbat and Takakuwa [2] state that the fundamental objective of Six Sigma methodology was the implementation of a measurement-based strategy that focuses on process improvement and variation reduction. Design for Six Sigma (DFSS) is a systematic methodology utilizing tools, training and measurement to design products and processes that meet customer expectations at a Six Sigma quality level.

The author discussed two Six Sigma sub-methodologies: DMAIC (define, measure, analyze, improve, control) and DMADV (define, measure, analyze, design, verify). Simulation models for improving the current conditions of the process use the DAMIC methodology. The development of an optimized "to-be" model uses DAMDV. The case study was conducted at the Ervei Khuder Mining Iron Production Factory (MIPF). As the main problem was process lead time, the goal was to increase factory capacity and remove bottlenecks.

The process at MIPF consists of two parts: mining and iron production. These two types of production can be treated as two sub-models. The mine sub-model includes drill, blast, load and transport utilizing truck haulage. The outcome of mine sub-model was to satisfy the demand of the iron production sub-model, which contained four processes in series:

i)The ore was crushed by two crushers (primary and secondary crusher)and then screened.

ii)The third crusher crushes the ore to a fine ore.

iii)The magnetic ore is separated from the fine ore.

iv)The separated usable iron is cleaned by two machines and sent to the endpoint.

Collected data were analyzed by an Arena Input analyzer to perform distribution fit analysis of input parameters. An "as-is" model was built to simulate the existing process. The model was then verified and validated.

One important part of this Six Sigma project was to establish the Critical-to-Quality (CTQ) characteristics as performance metrics. Chinbat and Takakuwa [2] decided that non-value added times such as wait time, truck utilization were the main metrics for this simulation model. The simulation model was altered to develop the "to-be"models. The "to-be" models were then evaluated by comparing the performance metrics to select the "best" options available for meeting the goals to increase factory capacity and diminish bottlenecks.

Chinbat and Takakuwa [2] summarized the importance of the role simulation played in this Six Sigma project:

i)Senior management noticed that the simulation provided the Six Sigma team with a clear view of the process bottlenecks and problems and the ability to visually understand the optimization with a numerical demonstration.

ii)The use of Arena software for the OPS (Operation Process Simulation) for optimization of current Mining Iron Production Factory (MIPF) and designing the new MIPF allowed for the ability to measure, analyze, improve and design the process with minimal post design rework, with realistic outputs.

2.2.2 Concurrent Simulation and Optimization Models for Mining Planning

Fioroni *et al.* [4] present a case study of how simulation and optimization models were combined to achieve a feasible, reliable and accurate solution for a short-term mining plan. Discrete event simulation tools are not suitable for directly solving complex optimization problems like a mining plan. In this research, the authors combine an optimization model with a simulation model to solve a mine planning problem. The

objective was to provide a feasible solution for quality and quantity production of ROM (Run of Mine) iron. The author further discussed, in two separate parts of the project, the simulation and optimization model.

The ROM (Run of Mine) iron is described by authors as follows: each mining area has used a truck/shovel combination to transport ore. Ore was carried to the main stockpile and primary crusher. Each mining area had a different grade of iron ore; therefore the correct quantity of ore in each area must be extracted to reach the desired quality (grade) and quantity at the ROM stockpiles. In other words, the mill receives ore from several mines, which deliver different grades of ore, and stores the ore in stockpiles. To meet output requirements or customer demand, a certain amount of ore should be pulled from the stockpile and fed to the mill circuit for processing. The real ROM iron has some variable effects on the system: loader maintenance, truck maintenance, exhaust of ore/waste at the mining area, etc. Averages do not precisely reflect truck operation's loading and transportation variations. Discrete event simulation is a suitable tool to reproduce this randomness and to evaluate the feasibility of the plan.

The optimization part of this problem focused on truck allocation and determines if truck fleets are assigned to correct areas. It estimates the number of trips for each truck to provide the desired ore grade at the ROM pile.

After explaining the simulation and optimization models, the authors outlined how these two models were combined in a run: "The simulation model initially runs with the optimized plan from the optimization model. The simulation runs until the system state changes due to an "event" such as equipment failure or truck maintenance. After the "event" happens, the optimization model is called and recalculates the new optimized plan. The simulation model continues with this new plan until the next "event" happens." This approach is a good integration of both tools instead of a post- or pre-simulation run. The number of interactions between simulation and optimization depends on the mining area number, the qualities, and total mass of ore; it also depends on the shovel and truck fleet availability.

The simulation model was developed in Arena using Visual Basic Application

(VBA) for communication between simulation and optimization models. The objective of this simulation model was to test the viability of the simulation of the mining plan proposed by the optimizer, presenting utilization and production. The simulation model also included equipment breakdown and re-planning or extraction in several areas in its evaluation.

The authors concluded that the project's objective was achieved by reducing mining costs through a combination of simulation and optimization. For daily planning, it supplied a tool which allow the decision maker to discuss the mine plan prior to execution; increase the trustworthiness of the mine plans; allow equipment utilization analysis; and make possible the analysis of several scenarios within a short time interval.

2.3 Mill Optimization Model

2.3.1 Economic Optimization of an Ore Processing Plant with a Constrained Multi-Objective Evolutionary Algorithm

The principle parameter used in most optimization model is profit, which combines minimizing investment and operating costs and maximizing return on investment, throughput and efficiency. Hubandi *et al.* [9] demonstrated, using multi-objective evolutionary algorithms (EAs) in a case study, an approach to optimize the design of a comminution plant. The two real-world complications are included in this model: risk management and complex feasibility conditions.

In the case study, the author defined the comminution circuit as a collection of physical processes that can be applied to a stream of ore to reduce the size of the particles in the stream. It includes primary, secondary and tertiary crushers; primary and secondary screens; coarse ore stockpiles; fine ore bins; milling and leaching. The milling lines in this case study have a pre-defined order in which they are utilized. Material is fed into the preferred line until that line reaches capacity, after which excess material is fed into the subsequent line(s).

The design parameters in this case study include feed variation, placement and

types of mills, and the feed order of the milling lines. There are four independent objectives were used to evaluate the overall system design.

The solution procedure used an evolutionary algorithms to solve the problem. As a result, it identified a number of operational changes that could add significantly to total earnings over the life of the project without any additional investment. At the same time, the optimal solution also fully utilized available crushing plant hours. However, the gains in overall project value achieved were relatively modest compared to the original plant design and operation.

2.3.2 Crushing Plant Optimization by Means of a Genetic Evolutionary Algorithm

The objective of plant modelling is to make a plant runs as efficiently as possible. Within a crushing plant it is difficult to know how each individual production unit affects the efficiency of whole plant. It is difficult to predict how a change in one particular parameter will influent the overall system's performance. Svedensten and Evertsson [20] developed software in Visual C++ to solve this type of problem. It has a graphical interface for crushing plant modelling and uses a "drag and drop" concept. Optimization parameters for various units can be easily selected via the dialog boxes.

The authors described the crushing plant as an unknown black box which consists of a configuration of the plant and mathematical constraints. Design parameters were input into the "the crushing plant" box and the output is a cost function of input parameters. The crushing plant model consists of four major parts:

i) Production unit: describes the transformation of the rock material as it passes through the unit.

ii) Rock material model: contains information about the rock properties that affect production.

iii) Economic model: cost function for evaluating the performance of the crushing plant from the perspective of the owner.

iv) Customer demand model: the requirement to generate an adequate level of product quality. User is able to adjust the levels of over- and undersize percentages of a product to accommodate customers' demand.

The author further discusses how the production unit can be broken down into more specific units such as crushers, separators, transporters, and storage. Storage consists of stockpiles and storage bins. Each of these units has a number of variable parameters that can vary between upper and lower limits, and static parameters. A variable can be optimized during a computer simulation run, or fixed, and therefore not included in the optimization.

The economic model contains a cost function to evaluate each production unit. The cost can be divided into two types: cost per ton (variable cost) and cost per hour (fixed cost). When a simulation runs, these two costs in every unit are calculated as a single cost per ton $C_{tot,i(X)}$ using the total mass of rock $m_{rock,i(X)}$ material pass through the unit i per hour, as follows where x is a input vector:

$$C_{tot,i(X)} = C_{ton,i} + C_{houri}/m_{rock,i(X)}$$

The total cost per ton is then accumulated against the rock material model and sent to the next production unit mode in the crushing plant model.

The objective of the model is to maximize the gross profit of the crushing plant. The solution must meet the customer's demand constraints as well. There are two types of constraints in the optimization: 1) some production units have a combination of optimization parameters that cannot be used due to incompatibility; 2) constraints related to product quality. The simulation optimization contains two components: a subroutine for generating parameter values and a simulation subroutine for evaluating the values generated.

The author concluded that crushing plant optimization model not only includes the performance of the different production units but also the economic aspects.

2.4 MIP of Mine Production Planning

Kuchta *et al.* [12] noted that mathematical programming has been used for production scheduling in underground mines. While heuristic algorithms may produce usable schedules, they do not necessarily find the best schedule because the solution is compared to the current modus operandi. Mixed-Integer Programming (MIP) has been used to generate an optimized production schedule. However MIP can be time-consuming due to the large number of decision variables, with many of them restricted to assume integer values. The case study done by Kuchta *et al.* [12] demonstrated a new mixed-integer programming model developed for the Kiruna Mine.

Kuchta *et al.* [12] described LKAB's Mine as having two types of ore bodies and delivering three types of ore product based on phosphorous content delineated as B1, B2 and D3 products. The mine provides one post-processing mill with B1 ore, two mills with B2 ore, and one mill with D3 ore. Electric Load Haul Dump (LHD) units transfer ore from the production drift to the ore passes, which extends further down to the transportation level. Then the ore is transported to the underground main crusher by a large train and hoisted to the surface.

The MIP model determines the starting date for various machine placements such that the required B1, B2 and D3 ores can be produced each month to meet demand. In other words, the mining production schedule must provide a mining sequence that takes into account the physical limitation of the mine and meets the required quantities of each raw ore type for each time period.

The objective of the model was to minimize the production deviation for each type of ore during each month. The operational constraints are described by authors as follows:

i) Accounting constraints that track the amount of each ore type mined on each month.

ii) Vertical and horizontal sequencing constraints that preclude mining a machine placement under a given machine placement until at least 50% of the given machine placement has been mined.

iii)Shaft group constraints that restrict the number of LHDs active within a shaft group at any one time to a predetermined maximum, usually two or three.

The MIP is implemented using the AMPL programming language and the CPLEX solver 7.0. The first version of this problem required 50,400 integer variables.

In the second version, Kuchta *et al.* [12] replaced the binary variable indicating whether the production block is mined in time period t with a binary variable indicating whether a machine placement starts to be mined in time period t. In this case, about twelve production blocks can be aggregated into a single machine placement, using both an earliest-possible start date and a latest-possible start date for each machine placement. With this reformulation, the number of integer variables is then reduced to 650 and a complete five-year schedule can be obtained in 300 seconds. The author explained that the number of integer variables can be further reduced by assigning earliest- and latest- possible start dates to machine placements based on the sequencing logic and demand constraints and bounds on a reasonable deviation between demand and production.

The implementation of this model brings major advantages to a company. Compared to previous manual scheduling, it was estimated that cost savings for the time spent generating schedules was about 25% of the scheduler's salary. In addition to this, mine planners also have the ability to compare various production schedules so that they can plan for changing demands or other contingencies.

Chapter 3

Formulation of Integer Programming Model to Calculate Draw Scheme and Campaign Schedules

The purpose of the Integer Programming (IP) model is to help plan the mill production campaigns and to define a proper draw scheme for each campaign period. The "Draw Scheme" specifies the type and proportions of ore that the mills draw from the fine ore bins. The draw scheme is constant during a given period. A null draw scheme indicates a shutdown of the mill. "Campaign" means a consecutive number of days which use the same draw scheme. The timing of switch-over from one draw scheme to another, which is also called "Campaign Switch-Over", is calculated by the IP model. The shortest period of time to be considered in this model is best thought of one day. If different time interval is desired, this can be substituted.

The outputs of the model are: 1) the length of each campaign; 2) the draw scheme to be used for each particular campaign. The overall objective of the IP model is to allow the decision maker to calculate a set of campaigns that minimizes some combination of 1) excess inventory levels of unprocessed ore; 2) the number of days on shutdown 3) the number of active piles required at any point in time.

3.1 Blending Model

A draw scheme amounts to a way of blending ores to proceed through the mill. This blending can occur either outside the mill, in the ore piles, or inside the mill via the fine ore bins feed to the mills. For each defined draw scheme, it has a precalculated draw rate (D_{ot}) for each type of ore. The draw rate is used to calculate total draw based on number of time units had for this draw rate. The draw rate (D_{ot}) and blending is influenced by the following: 1) For a copper (or nickel) ore, the mills have a nominal feed rate for each ore at a nominal grade of Cu%(Ni%). If the total metal content

of the ore feed exceeds this percentage, the rod mill can be throttled to maintain a metal feed rate equivalent to the rate at the nominal Cu%(Ni%). However, the mill can only be throttled back to about 75% of the nominal feed rate. Alternatively, ores can be blended by using two fine ore bins to feed the mill. The principle of blending is to have an equal metal content of Ni and Cu in the blending process. The blend ore is treated as a Ni ore for feed rate purposes and throttling back of the mill, if necessary, is as if the feed were Ni.

A small linear programming model has been developed that can calculate a blending formula and corresponding draw scheme for any combination of input ores. The main idea of this model are: there are different types of ore received by mill. Each type of ore has Cu and Ni grade. If all these ores can be allocated to proper FOB for blending purpose without exceeding the maximum number of blender in mill system, it is not necessary to perform pre-blending in the ore piles. The objective of this blending model is to minimize the total number of ore blending in FOB. For this particular problem, since there are two grinding circuits exists, if number of blending is less than two, all blending can be done in the mill. The solution from this blending model provide a blending formula of how much ore should be allocate to each FOB by obeying the equal metal content rule.

3.1.1 Formulation of Blending Model

The model is developed by using GUSEK and showed as followed:

Sets:

Blender	Set of blenders
Ore	Set of Ores

Indexes:

$o \in Ore$	Type of Ore

$b \in Blender$ Blender

Parameters:

nOre	number of different types of ore
nBlender	number of blender in the mill system
$CuGrade_o$	Copper grade of ore type o
$NiGrade_o$	Nickel grade of ore type o
FOBRatio	Maximum blending ratio between FOBs
MinNi	The minimum output of nickel from the system

Decision Variables:

x_{ob}	amount of ore type o in blender b
y_{ob}	binary variable, $y_{ob} = 1$ means ore types o is blended in blender b.
z_b	total amount of ore in blender b.
$CuContent$	total copper content in all blenders.
$NiContent$	total nickel content in all blenders.

Objective Function:

$$Minimize \quad \sum_{o \in Ore, b \in Blender} y_{ob} \tag{3.1}$$

Subject to:

$$x_{ob} \leq y_{ob} \quad o \in Ore, b \in Blender \tag{3.2}$$

$$\sum_{o} x_{ob} = z_b \quad o \in Ore \tag{3.3}$$

$$\sum_{b} z_b = 1 \quad b \in Blender \tag{3.4}$$

$$CuContent = \sum_{b} x_{ob} \times CuGrade_o \quad b \in Blender \tag{3.5}$$

$$NiContent = \sum_{b} x_{ob} \times NiGrade_o \quad b \in Blender \tag{3.6}$$

$$CuContent = NiContent \tag{3.7}$$

$$z_{[b]} \leq FOBRatio \times z_{[b+1]} \quad b \in Blender \tag{3.8}$$

$$z_{[b+1]} \leq FOBRatio \times z_{[b]} \quad b \in Blender \tag{3.9}$$

$$NiContent \geq MinNi \tag{3.10}$$

$$NiContent \leq MaxNi \tag{3.11}$$

The meaning of each constrain is:

(3.18) If there is any amount of ore in blender b, blender b should be used;

(3.19) summation of different ore type in blender b should be equal to total amount of ore in blender b;

(3.20) It was assume there is one tonne of ore are processed in the system. Summation of all ores in all blenders should be one tonne;

(3.21) Calculate the copper content in all blenders;

(3.22) Calculate the nickel content in all blenders;

(3.23) Equal metal content of Ni and Cu;

(3.24,3.25) Make sure the maximum blending ratio between two FOBs are achieved;

(3.26,3.27) Total nickel content in all blenders should at least equal to the desired output of nickel.

3.1.2 Numerical Results of Blending Optimization Model

Formulation of a sample problem in GUSEK is showed in App.D and solution for the this problem was found within .218 seconds. The output solution file generated from GUSEK showed in App. E.

The solution generated from this blending model can be considered as a blending formula which defines several specifications: 1) y_{ob} defines which type of ore should be blended in specific blender b; 2) x_{ob} defines how one tonne of ore are allocated to different blenders according y_{ob}.

The blending formula is part of campaign draw development but only focuses on the control of the draw scheme at the blender and FOBs. The campaign scheme discussed earlier has a wider control scheme for the overall system.

3.2 Mill Optimization Model

The mill optimization model works as follows: mill receive different types of ores from dummy mines. Arrival rate of each ore varies daily. This accommodate the uncertainty from mines production. Once mill receives ores, it stays at stockpiles. Each stockpile can only store one type of ore at a time. It is possible to have a small number of active stockpiles outside of mill. The mill starts to pull ore from stockpiles based the campaigns and draw scheme solution. The draw scheme defines the draw rate for each type ore and campaign defines how long should this draw lasts. This type of campaign is defined as operation campaign with a campaign length constraint. Campaign schedules should keep the inventory level of each ore as low as possible. At the same time, meet the draw requirement for each draw scheme. Shutdown

campaign is a special campaign with zero draw rate. During shutdown, it can be performed maintenance tasks or waiting for ore coming. It is encourage to have as many shutdowns as possible during total planning horizon.

3.2.1 Mill Optimization Model Parameters

The optimization model's main data includes:

- Forecast ore arrivals in each day;
- A list of potential draw schemes;
- Draw rate of each ore type for each draw scheme in each day;
- Maximum and minimum total inventory limits;
- Maximum and minimum inventory of each type of ore;
- The weight applies to the objectives: γ and β;

3.2.2 Mill Optimization Model Formulation

The optimization model is detailed below:

Sets:

T	Set of days
D	Set of draw schemes
O	Set of Ores
H	Number of days of planning horizon

Indexes:

$k \in D$	draw scheme used for particular campaign. $k^{SD} = 0$ means shutdown
$o \in O$	Particular type of ore
$i \in H$	campaign starts at the beginning of day i
$j \in H$	campaign ends at the beginning of day j
$t \in T$	time period in day t

Parameters:

N	number of campaigns to be used for planning horizon
L^{max}	maximum campaign length in days
L^{min}	minimum campaign length in days
I_{tot}^{max}	maximum total inventory allowed on day t
I_{tot}^{min}	minimum total inventory allowed on day t
I_{ot}^{max}	maximum inventory allowed for ore type o on day t
I_{ot}^{min}	minimum inventory allowed for ore type o on day t
A_{ot}	the arrival of ore type o on day t
D_{ok}	the draw rate for ore type o under draw scheme k
NP^{max}	maximum number of active piles allowed on day t
γ	weight of parameter z^{SD} (indicates the relative importance of number of shutdown during campaign period)
β	weight of variable V_{NP} (indicates the relative importance of number of active piles)

Decision Variables:

x_{ij}	binary variable, $x_{ij} = 1$ means campaign starts at the beginning of day i and ends at the beginning of day j
y_{ijk}	binary variable, $y_{ijk} = 1$ means draw scheme k is used during

	campaign day from i to j
d_{ot}	draw of ore type o on day t
i_{ot}	ending inventory of ore type o at the end of day t
V_t^+	measure of the violation of the upper bound on total ending inventory occurring on day t
V_t^-	measure of the violation of the lower bound on total ending inventory occurring on day t
v_{ot}^+	measure of the violation of the upper bound on ending inventory occurring on day t for ore type o
v_{ot}^-	measure of the violation of the lower bound on ending inventory occurring on day t for ore type o
z_{ot}	binary variable, $z_{ot} = 1$ means ore type o pile is active on day t
V^{NP}	maximum number of active piles on each day t
z_{SD}	number of days on shutdown during the planning horizon

Objective Function:

$$\textit{Minimize} \qquad \sum_{t=1,t}(V_t^+ + V_t^- + \sum_{o\in O}(v_{ot}^+ + v_{ot}^-)) - \gamma z_{SD} + \beta V^{NP} \tag{3.12}$$

The model is considered as weighted goal programming problem. The objective of this model are: 1) minimize the excess total inventory level and inventory level for all types of ores; 2) maximize the number of shut down days during the planning horizon; 3) minimize the number of active piles. Because it is difficult to attain all these goals at the same time, then the weight factor γ and β in the objective function are used to encourage more shutdown days by increasing γ or to discourage active piles by increasing β. The user can assign a different weight to each specific objective. For example, if γ is greater than β, it means that having more shut down time is more important than reducing the number of active piles in overall optimization.

Subject to:

Campaign definition constraints:

$$x_{ij} = 0 \quad \forall \quad j - i > L^{max} \tag{3.13}$$

$$x_{ij} = 0 \quad \forall \quad j - i < L^{min} \tag{3.14}$$

$$\sum_{i=1}^{H-1} \sum_{j=i+1}^{H} x_{ij} = N \tag{3.15}$$

$$\sum_{m=1,i-1} x_{mb} = \sum_{n=i+1,H} x_{bn} \quad \forall \quad b = 2, H-1 \tag{3.16}$$

$$\sum_{j=i+1}^{H} x_{1j} = 1 \quad \forall \quad i = 2, H-1 \tag{3.17}$$

$$\sum_{k} y_{ijk} = x_{ij} \tag{3.18}$$

$$z_{SD} = \sum_{i=1,H-1} \sum_{j=i+1,H} (j-i) y_{ijk^{SD}} \tag{3.19}$$

Material flow and inventory constraints:

$$d_{ot} = \sum_{i \le t} \sum_{j > t} \sum_{k} D_{ok} y_{ijk} \tag{3.20}$$

$$i_{ot} = i_{ot-1} + A_{ot} - d_{ot} \tag{3.21}$$

$$I_{tot}^{min} \le \sum_{o=1,O} i_{ot} - V_t^+ + V_t^- \le I_{tot}^{max} \quad t \in T \tag{3.22}$$

$$I_{ot}^{min} \le i_{ot} - v_{ot}^+ + v_{ot}^- \le I_{ot}^{max} \quad t \in T, o \in O \tag{3.23}$$

$$d_{ot} \geq 0 \quad i_{ot} \geq 0 \quad v_{ot}^{+} \geq 0 \quad v_{ot}^{-} \geq 0 \quad V_t^{+} \geq 0 \quad V_t^{-} \geq 0 \tag{3.24}$$

Active piles constraints:

$$i_{ot} \leq (\sum_{t=1,t} A_{ot}) * z_{ot} \tag{3.25}$$

$$d_{ot} \leq (\max_{k}(D_{ok})) * z_{ot} \tag{3.26}$$

$$\sum_{o=1,J} z_{ot} \leq NP^{max} + V^{NP} \tag{3.27}$$

The meaning of each constraint is:

(3.2) Campaign starts from the beginning of day i and ends at the beginning of day j. The length of campaign j-i should be smaller than the maximum length of campaign allowed L^{max};

(3.3) The length of campaign (j-i) should be greater than the minimum length of campaign allowed L^{min};

(3.4) total number of campaigns required for planning horizon should equal the pre-specified N.

(3.5) The number of campaigns starting from day t equal the number of campaigns that end at the beginning of day t;

(3.6) One campaign starts on day 1;

(3.7) Only one draw scheme can be used in each campaign;

(3.8) The number of shutdown days is the sum of the days in those campaigns that use the shutdown campaign type k^{SD};

(3.9) The draw on day t for ore o is equal to the draw rate of the draw scheme k that is in use during the campaign active on day t;

(3.10) Ending inventory on day t for ore o is equal to the ending inventory of ore o of the previous day plus the arrival of ore o in day t minus the draw of ore o;

(3.11) At the end of day t, the total inventory should be between lower and upper bound; V_t^+ and V_t^- means the violation of these bounds;

(3.12) At the end of day t, the inventory of each ore type o should be between the specified upper and lower bounds for that ore type on day t;v_{ot}^+ and v_{ot}^-means the violation of these bounds;

(3.13) Draw and inventory level should be greater than zero;

(3.14) Ending inventory of specific ore pile o should be less than cumulated arrival of ore type o if pile is active; z_{ot} indicate that ore pile of type o is active;

(3.15) draw from a pile should be less than maximum draw rate over all draw scheme if pile is active;

(3.16) total number of active piles should be less than the maximum number of active piles allowed in any period. The variable V^{NP} measures the violation of this limits.

3.2.3 Model Implementation

The open source GNU [6] Linear Programming Kit (GLPK) [5] has been used to implement the model. It is a software package intended for solving large-scale linear programming , mixed integer programming, and other related problems, and is a part of GNU project. GUSEK [8]provide an open source LP/MILP IDE for Win32, packing a custom version of the SciTE [18] editor linked to the GLPK standalone solver.

3.2.4 Numerical Results of Mill Optimization Model

The mill optimization model was developed using the GUSEK code as shown in App.A. The model used external .dat files as its input file and generated an output file when the model was successfully processed.

The external data file, as showen in App.B summarizes the model input as shown below:

i)Total length of planning horizon is 20 days with at most 4 campaigns required for this time period.

ii)The length of each campaign ranges from minimum of 2 days to maximum of 7 days.

iii)There are three types of ores being processed.

iv)There are four blending draw schemes available to be chosen for a campaign including one shutdown draw scheme.

v)The maximum number of active ore piles is five.

The model is able to find an integer optimal solution and generate the results shown in App.C. A separate detailed output file can be generated as well after the model has been successfully processed. The results for this sample problem indicate:

i)Campaign one: Day 1 to Day 6 using draw scheme 1;

ii)Campaign Two: Day 7 to Day 12 using draw scheme 3;

iii)Campaign Two: Day 13 to Day 17 using draw scheme 1;

iv)Campaign Two: Day 18 to Day 20 shut down;

It was observed that the mill optimization model has the capability to solve simple problems within a reasonable time. The proposed campaign and draw scheme generated from the model is the anticipated solution. By altering some input parameters of the model such as number of shutdown and number of campaigns, the model was able to solve the problem correspondingly. Further sensitivity analysis of input parameter was recommended.

Chapter 4

Development of Simulation Model

Theoretically, it should be possible to run a simulation with all potential campaign and draw scheme combinations to find mill production system that achieves the best throughput performance. However, this approach is not feasible since it leads to a very large number of simulation runs. The mathematical model does not capture the complexity of mill production system.

In this research, the objective is to develop a dynamic, stochastic discrete-event simulation model for a decision-maker to evaluate a mill process system with stochastic conditions. This discrete event simulation model has been adopted in order to evaluate the actual system performance for a specific campaign and draw scheme developed from the IP model.

Several important steps in the simulation development are included in this chapter: Section 4.1 discusses the problem of interest and system configuration; section 4.2 discusses the system structure and its operating procedures; section 4.3 explains the way to construct a model in Visual Basic for Application (VBA); and section 4.4 discusses output analysis. Data analysis will be discussed in Chapter 5. Simulation - based sensitivity analysis will be presented in Chapter 6.

4.1 Formulation of the Problem

4.1.1 Problem of Interest

This research focuses on the development of a suite of tools that will allow a decision maker to model and control the dynamics of ore receiving, sequencing and comminution operation of the mill to make better operational decisions, resulting in increased yields and greater margins. For a complex system, like a mill operation, simulation

is often the only feasible method to investigate the problem and evaluate alternative operational conditions. The characteristics of discrete-event simulation are as follows:

i) *Dynamic Simulation*: Instead of focusing on the representation of a system at a particular time, a dynamic simulation model represents a system over time.

ii) *Deterministic/Stochastic Combined Simulation*: The deterministic element of the simulation is material transfer. The transfer activity can be "determined" by input quantities, specific equations and certain transfer control policies. The mill model also has random elements such as machine failures which gave rise to stochastic simulation elements. Since stochastic inputs produce random outputs as well, simulation models can be treated as an estimate of the real system. This is the reason why verification and validation is critical to any simulation model.

iii) *Discrete Simulation*: Although, in general, continuous simulation is more appropriate for simulating material flow, in this research project, the simulation model concerns the material movement between "stations" that have characteristics. In other words, material flow is not only "determined" by defined equations, but more importantly depends on a series of control policies of a "station". For example, if a "station" is receiving status, a certain amount of material will be delivered to it. As a result, its current inventory variable updates and this "station" changes its status to delivery status. In such case, the simulation uses Discrete Event Simulation (DES) approach, in which the state of a system changes only with an event occurs at discrete points in time. Maintenance and machine failure in the real world are simulated as stochastic elements in the model.

The simulation model is implemented in VBA in Microsoft Excel instead of using a commercial simulation package. The advantages of using VBA are as follows:

i) *Flexibility*: Engineers usually have strong working knowledge of the Microsoft Office suite of products. It is thus easy for them to alter a set of

parameters in the simulation model to perform an experimental run and generate the results.

ii) *Relative low cost*: Excel is widely adopted within many organizations. A specific simulation software may require additional cost to purchase the software and training for the users.

iii) *Reusability*: the simulation source code can be used again by developers with solid knowledge of programming to add new functionalities with slight modifications.

4.1.2 System Configuration

There are two main issues associated with the system configuration: the level of model detail and the relationship between subsystems.

The level of detail included in the simulation model is dependent on the scope of the project. Because the entire mill operation involves many machines and processes, it is almost impossible to simulate them all. Likewise, it is very difficult to obtain data, such as failures for every machine. Consequently, it is important to set a level of detail that is neither too high (whereby the analysis may ignore important relationships between subsystems) or too low (which might result in spending too much time understanding precise configurations of a subsystem). Rather, the overall system should be partitioned into subsystems. Each subsystem should be defined so it can be reused, redesigned and is flexible. Block diagrams are typically used to show relationships between subsystems at a high level.

The core concept of this simulation model is to represent a subsystem as a "station". A subsystem in mill production system can be mine, crusher, surge bin or grinding circuit. It is possible to create several subsystem categories to accommodate these. An efficient way to define all different subsystems into one single module is implemented in simulation by introducing the "station" concept. In this case, the station is a module that receives input and delivers output to the next station. The connection between stations creates material flow for simulation purpose. All station

shares the same properties among all subsystem. The user is able to define these properties to distinguish station from each other. In the later section, a detailed discussion of how the "station" is coded as class module in VBA is presented.

One advantage of using the station concept is its flexibility to customize material flow. By understanding from the structure of the mill operation, we are able to define the station and its predecessor stations and successor stations. Material is then simulated as flow between "stations". The mill operation includes two phases: surface and circuit. Surface processes include mines, initial ore piles, and primary crushers. Material received from mines is stored at the initial piles and may be crushed by the primary crusher before being sent to the circuit. The circuit process includes the surge bin, crusher, fine ore bins and grinding circuit. Based on Figure 1.2 and 1.3, a block diagram of mill operation is developed by applying the station concept as shown in Figure 4.1 and explanations of each station are as follows:

i) *Mine 1-3*: Dummy mines which deliver ore to the mill.

ii) *Initial Piles 1-6*: store un-crushed ores.

iii) *Primary Crusher*: primary crusher crushes ore before mill process.

iv) *Stockpiles 1-6*: store crushed ore for mill process.

v) *Surge bin*:No.1 surge bin before the crushing process.

vi) *Crusher*: entire crushing process.

vii) *FOB 1-5*: fine ore bin in the grinding circuit.

viii) *Grinding Mill A,B and C*: three grinding circuits A, B and C.

4.2 Model Definition

4.2.1 Definition of Key Terms

This section provides short definitions of some of the central terms and, will link the discussion of key concepts.

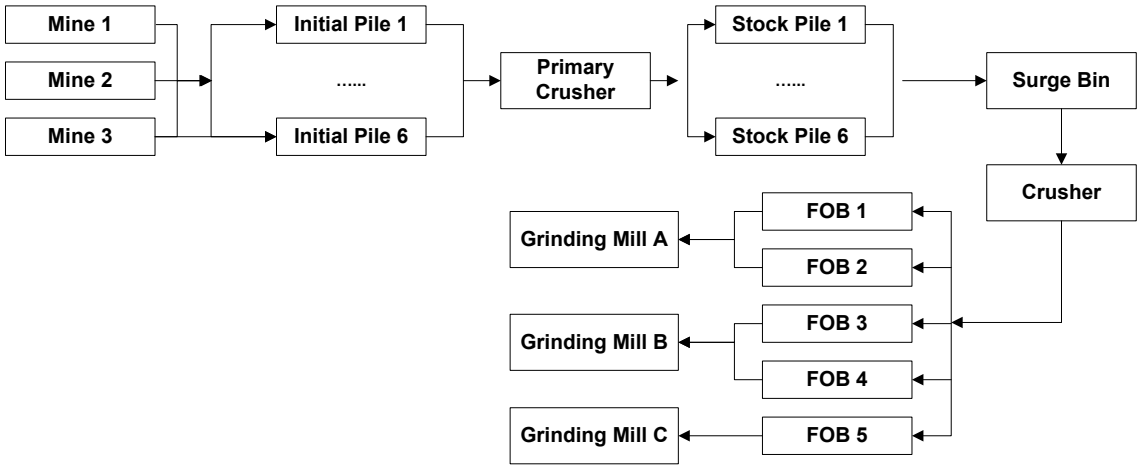

Figure 4.1: Block Diagram of Mill Operation

i) *Blender Scheme*: A "recipe" specifying, for blending, the draws of ore of a given type from one fine ore bin (or more than one) and their ratio (if mixing two types of ore).

ii) *Station Scheme*: consists of specifying, for all stations except blending, rock type, transfer rate, predecessor, successor, and so on.

iii) *Campaign*: a consecutive number of days which use the same "Campaign Scheme".

iv) *Campaign Scheme*: scheme combines the "Blender Scheme" and "Station Scheme".

4.2.2 Simulation Model Assumptions

Before discussing the operational procedures of the comminution portion of the mill, it is important to ensure that the model's assumption are correct and complete. This helps to focus the project objective and avoid significant reprogramming later. The assumptions are as follows:

i)It is assumed that production from the mine has no effect on the mill process. In other words, the "Mine Station" has infinite material which its

successor can draw from and the availability of its ore type is not constrained. There are three individual mines in the model which can deliver up to three different types of ore into the mill within one campaign period. Mines also can be modified as receiving ore according to specific ore delivery schedule from mines which is similar to the ore arrival rate in the mill optimization model. This delivery schedule can be further linked to the mine production schedule.

ii)There are six initial piles and six stockpiles in the model, each of which is a storage place for any type of ore. In reality, it is better to keep a small number of active stockpiles. The optimization on usage of these piles is not included in the simulation.

iii)There are six pre-defined types of ore in the model: low-grade of Ni and Cu, medium-grade of Ni and Cu and, high-grade of Ni and Cu.

iv)There are five Fine Ore Bins in the model. The rock type in the FOB is defined in the "Station Scheme".

v)There are three blenders (A, B and C) in the model, and their properties are defined in the "Blender Scheme".

vi)In order to match the real-world system, the simulation model is constrained by which of the 5 fine ore bins is used to feed the grinding mills; FOB 1 and 2 feed grinding rod mill A; FOB 3 and 4 feed grinding rod mill B, and FOB 5 feeds grinding ball mill C.

vii)There are no screening systems in the simulation model. The particles from the crusher are assumed to always pass the screening system without additional re-crushing. In the real world, large rocks are sent to a secondary or tertiary crusher to meet the desired size of the particle. In the simulation model, this could easily be done by assigning a certain percentage of product to go to a re-grinding process and the rest to go to the next station in the process. Since not enough data is available about screening efficiency, the re-grinding process is not included in the simulation.

4.2.3 Operating Procedures

Operating procedures are critical for the simulation model. They transfer real-world operational policies into model logic. The material flow control logic in the simulation model is key to making the model run. In general, it defines two things: when a receiving or a delivering event happens and the amount being transferred. During a simulation run, each "station" follows its operating policy and continuously evaluates certain variables to update the "station" state or to execute an "event". There are many types of "stations" in the model, some have similarities and some are unique. The operating procedures for "station" are described as follows:

i) *Mine Station*: Since the mine is the initial starting point of the model, it can only deliver material downstream to the initial pile. It delivers to the specific initial pile according to the type of ore extracted at the mine.

ii) *Initial Plies and Stockpile Stations*: Each pile is allocated to store one specific type of ore according to the rock type parameters defined in the "Station Scheme".

iii) *Crusher Station*: the crusher can only process one type of material at a time. Before receiving the next type of material, the crusher has to be emptied.

iv) *Surge Bin*: The inventory control policy plays an important role in executing the receiving and delivering action of an event. Like the crusher, the surge bin can only store one type of material at a time and must to be emptied before receiving a different type of material.

v) *Fine Ore Bin Station*: In a "push" system, the unique policy for the FOB is its emptying strategy. The model first compares the number of ore types in the current system with the number of ore types coming for the following campaign. It decides whether some FOBs need to be emptied or not, when the current system can not accommodate all different types of ore from next campaign. Then it decides which FOB(s) in the current campaign

has (have) to be reduced to zero inventory to accommodate the additional types of ore. The FOB with the least material is selected to be emptied first. In a "pull" system, quantities of material required to meet demand is pre-calculated so that there will be no material left in the FOB at the end of each campaign.

vi) *Grinding Mill Station*: The grinding mill follows the blender scheme as its operating policy. It pulls material from the FOBs and mixes them according to a defined ratio to achieve an acceptable grade of feed to the mill flotation circuit.

Besides stations, general control policies applied to the simulation model to ensure material flow through the entire system:

i) *Crushing Circuit*: The crushing circuit includes two stations in the simulation model: the surge bin and the crusher. The crushing process control policy forces these two stations to process the same type of ore. At anytime, if crushing need to process a different type of ore, these two stations must be both emptied before the new ore comes. In the simulation model, the ore type in the crushing process is defined as "global ore type". When ore waiting to be processed for crushing process differs from the global ore type, nothing can enter the crushing process.

ii) *Global Ore Type Switch*: To perform a switch of the global ore type for the surge bin, the global ore type is updated, if necessary, when the surge bin cycles. At every time step, a subfunction is called to determine the global ore type. It calculates the run-out time, including the time to empty both the surge bin and the crusher, and then sets the ore type with the lowest run-out time to be the global ore type.

iii) *FOB and Grinding Mill*: The FOB is a special station in the simulation model. It is associated with a grinding mill. The simulation model has the flexibility for a grinding mill to pull ore from two FOBs with two different types of ore at any blending ratio. These configuration are defined in

blending scheme. The amount of ore that can be transferred is determined by the ore type and inventory level of the grinding mill.

4.3 Construction of the Simulation Model

In this section, the modelling approach, details of the programming code, and the input and output interface are discussed.

4.3.1 Modelling Approach

The mill operation is modelled by using two main concepts: *Stations and Blenders*. A *station* is a class object which takes in material that has been pushed to it from an upstream station and outputs this material to a downstream station according to certain control rules. A *blender* is another class object that takes material from one or more upstream stations in fixed proportions and pushes it to a downstream station.

The simulation model runs as a "pull system" and is a customer-demand-driven model. In this case, the last station blender pulls material from its upstream FOB according to the Blender Scheme; then the FOB pulls material from its upstream crusher. The whole process continues until demand is met.

When the process map of material flow is defined by using station and blender concepts, the next important element is the modelling approach. There are two principal approaches that have been suggested for simulation model: *next-event time-advance* and *fixed-increment time-advance*. Since the fixed increment is a special case of the first one, next-event time-advance approach can be used for all discrete event simulation models, including the mill model.

Kelton and Law [10] summarized next-event time-advance approach as followed: "the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of the occurrence of the *most imminent* (first) of these future events, at which point the state of the system is updated to account for the fact that an event has occurred,

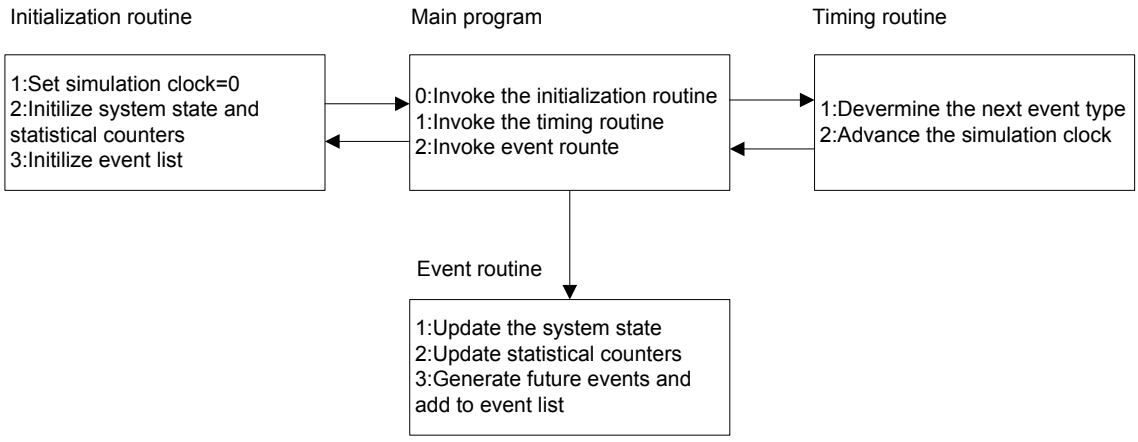

Figure 4.2: Flow control for the discrete-event simulation

and our knowledge of the times of occurrence of future events is also updated. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc." The time from one event to another is skipped in next-event time advance approach. In this case, the simulation clock jumps from one event to the next event. The standard flow control for next-event time-advance approach is shown in Figure 4.2.

Kelton and Law [10] described fixed-increment time-advance approach as follows: "the simulation clock is advanced in increments of exactly Δt time units for some appropriate choice of Δt. After each update of the clock, a check is made to determine if any events should have occurred during the previous interval of length Δt. If one or more events were scheduled to have occurred during this interval, these events are considered to occur at the end of the interval and the system state (and statistical counters) are updated accordingly." The fixed-increment time-advance can be realized when using the next-event time-advance approach by artificially scheduling "events" to occur every Δt time units.

One key element of fixed-increment time-advance approach is the Δt which is defined as *time step* in the mill model. At the end the each time step, it will have only one event "waiting to be executed". After this event is executed, several system states are updated according to the execution of that event. The primary use of this approach is for systems where it can reasonably be assumed that all events actually

occur at one of the times $n\Delta t$ for an appropriately chosen Δt. Thus, this approach is more appropriate for approximating a system that system states changes continuously over time.

The mill model is now illustrated in detail using the fixed-increment time-advance approach: All events actually occur at the end of one time step. To prevent simulation deadlock, the entire simulation model is run from back to front, which is similar to the pull system operating policy. In this case, the simulation model starts from the last station, which is the grinding station. If the grinding process predecessor, which is the FOB station, has enough material to transfer to grinding station and both grinding circuit and FOB stations meet the requirements for transferring (such as inventory level limits and status of machine), then this transfer event is assumed to take place. Otherwise, the grinding process will not pull material from FOB station. If FOB does not have enough material, it will pull material from its predecessor and repeat at each time step until it has enough material. In general, all stations pull material from their predecessors instead of pushing to successors, so that all stations receive material from their predecessors before they send material out. This is the key point to prevent simulation deadlock.

4.3.2 Implementation in VBA

Visual Basic for Applications, or VBA, is a computer programming language that is used as a macro language for all Microsoft products, including Excel. VBA controls Microsoft Excel by means of macros which are also called procedures. The mill simulation model contains two parts: the simulation run module and reporting routine. The simulation is programmed by following the next-event time-advance approach. The reporting routine summarizes data and presents them in charts, that are generated after the simulation run.

Module

Two types of modules are widely used in the programming code. *Standard Modules* store procedures and functions and can be either Private or Public. It can be accessed

either from within that module only (Private) or from anywhere in the project (Public). Standard modules are used to store variables, constants and declarations that needed in the project. This availability makes it easy to split the entire procedure into different modules for organization and ease of maintenance. The mill simulation model was broken into several standard modules as follows:

i) *modSim*: contains initialization, timing routine, statistics collection and the main simulation run. It serves as the base for simulation models which interact with all other modules.

ii) *modCampaign*: a module that deals with transition between campaigns.

iii) *modDowntime*: simulates machine failure events according to Mean Time Between Failure (MTBF) and Mean Time to Repair(MTTR) by using a next-event time advance approach.

iv) *modDisplay*: updates every time-step to display outputs from the simulation.

v) *modADO*: creates a statistics report and bar charts at the end of the simulation run.

vi) *Variates*: generates random variates from probability distributions.

vii) *modSchedules - optional*: specifies the production schedule of the mine, which includes operating time and production volume.

viii) *modTrucking - optional*: uses trucks to transfer material between some stations.

A detailed discussion of a sample "Standard Module" is given in App. F.

Class Module is another type module used frequently within the simulation model. It allows for the creation of objects which can have their own properties and methods like any other object. There are three major class modules, defined as follows:

i) *Rock Type Class Module*: the entity of the simulation model. The entire process is described as "Rock" flows through stations. Properties associated with rock class modules are records like "attributes" during the simulation run, and are listed as follows:

- Run-out time of the rock type: is the time to clear the rock type from entire system, also called pipeline time.
- Current rate at which this rock type is being consumed.
- Volume in the pipeline/surge bin attempting to be put into the FOBs.
- Current volume of rock type in the system.

ii) *Station Class Module*: takes in material that has been pushed to it from an upstream station and outputs this material to a downstream station according to certain control rules. Attributes associated with station are listed as follows:

- Station name and number.
- The rocktype the station is running.
- Maximum and minimum capacity level of station.
- Output rate in tonnes/time unit.
- Next station number defines successor and previous station number which defines its predecessor
- Whether the station is in working condition or not.
- Current station inventory level and previous (one time) step inventory level.
- The global rock type.
- Cumulative material flow through the station.

iii) *Blender Class Module*: takes material from one or more upstream stations(FOB) in fixed proportions and pushes it to a downstream station. In this simulation model, up to two different types of ore can be mixed in the

blender and are named as Rock Type A and B. Attributes associated to the blender are listed as follows. A detailed discussion of "Blender Class Module" is given in App. G:

- A collection of FOBs assigned to the mill.
- Type of Rock Type A and B.
- The percent of Rock Type A and B being mixed.
- The time to run out of Rock Type A and B.
- The rate of consumption of Rock Type A and B.
- The amount of Rock Type A and B to transfer.

Instead of using a single Class Module, a *collection* in VBA provides a convenient way to refer to a group of objects and collections as a single object. A collection is a series of single objects where each object shares the same characteristics and methods. In other words, all objects can be described the same way. In the mill model, collections are used widely to refer to group of similar items. Collections are manipulated by different *Methods* to perform different tasks. Standard methods includes adding/removing an object to/from a collection; counting the number of items in a collection and accessing an item in a collection. Several collection used in this model are described as followed:

i)station collection: A collection of stations that share same characteristics of "station" object. Sub collections in this category includes blender collection, FOB collection etc. For example, the FOB collection specifies which FOBs feeds material to which blender.

ii)rock-type collection: A collection of rocks. There are six different types of rocks grouped as one rock-type collection.

iii)campaign collection: refers to group of campaigns planned during planning horizon and are arranged in ascending order of starting time of the campaign. Once one campaign finished, it is removed from the collection.

Simulation Parameters	
Run Time	100
Time Step	0.25
Warmup Time	0
Num Reps	1
Animation	TRUE
Start Day	2009/1/1
Start Time	0:00:00

Figure 4.3: Simulation Parameters Input

4.3.3 Interface Design

The principle on which the interface design for the mill simulation model is based is that it should be simple to use. The entire simulation model is implemented in VBA rather than in a commercial specialized simulation software. Since the model focuses on the simulation of a process rather than the graphical presentation of a problem, the entire interface is simplified and includes two main sections: input parameter and output results.

Input Parameters

- Simulation Parameter Input (Figure 4.3)defines the control parameter to run the simulation model and includes “Run Time” (simulation replication length), “Time Step” (simulation advance step), “Warm-up Time”, “Num Reps” (number of replications), “Animation” (option to turn screen updates on or off), and “Start Day and Time” (simulation clock).

- Campaign Parameters Input (Figure 4.4) defines the parameter of campaigns planned during a simulation run. It includes “Campaign” (index of each campaign), “Draw Scheme Index” (index of draw and blender schemes used for this campaign), “Start and End Time” (start and end time of each campaign in

Campaign						
Campaign	Draw Scheme Index	Start	End	Mine 1	Mine 2	Mine 3
1	3	0	50	1000	1000	0
2	4	50	100	2000	3000	4000

Figure 4.4: Campaign Parameters Input

Draw Scheme	*Parameters*	*GrinderC*	*GrinderB*	*GrinderA*
1	RockType	1	1	1
	Target RockType Index	1	1	1
	Is Working?	FALSE	FALSE	TRUE
	Rate	100	100	100
2	RockType	1	1	1
	Target RockType Index	1	1	1
	Is Working?	FALSE	FALSE	TRUE
	Rate	100	100	100
3	RockType	1	1	1
	Target RockType Index	1	1	1
	Is Working?	FALSE	TRUE	TRUE
	Rate	100	100	100

Figure 4.5: Draw Scheme Input

hours), and "Mine 1, 2 and 3" (arrival from each mine during each campaign).

- Draw Scheme Input (Figure 4.5) lists several potential draw schemes a campaign can choose from. Each draw scheme specifies 1) the rock a current station has and its target rock type, 2) whether the station is in use or not, and 3) the station transfer rate. A new draw scheme can be added to the draw scheme list here.

- Blender Scheme Input (Figure 4.6) lists several potential blender schemes a campaign can choose from. It specifies which rock type of A and B is to be used for each blender, the percentage of rock type A to be mixed in each blender, the associated grinding mill for each blender, and the associated FOB for each blender. The blender scheme index is associated with the draw scheme index. Once the campaign selects the draw scheme index, it automatically selects the blender draw scheme index as well.

Draw Scheme	Parameters	BlenderA	BlenderB	BlenderC
1	Percent A	1	0.5	0.75
	RockTypeA	1	2	1
	RockTypeB	2	1	2
	Associated Grind Mill	GrindA	GrindB	GrindC
	Associated FOB	FOB1	FOB3	FOB5
	Associated FOB	FOB2	FOB4	
2	Percent A	1	0.5	0.5
	RockTypeA	2	2	1
	RockTypeB	1	1	2
	Associated Grind Mill	GrindA	GrindB	GrindC
	Associated FOB	FOB1	FOB3	FOB5
	Associated FOB	FOB2	FOB4	
3	Percent A	0.5	1	0.3
	RockTypeA	1	3	2
	RockTypeB	2	3	3
	Associated Grind Mill	GrindA	GrindB	GrindC
	Associated FOB	FOB1	FOB3	FOB5
	Associated FOB	FOB2	FOB4	

Figure 4.6: Blender Scheme Input

System Variables					
Replication	Clock Time	Date	Time	Rock Type	Bypass
1	45.25	Fri, 02-Jan-09	21:00:00	Med_Ni	FALSE

Figure 4.7: System Parameter output

Output Results

The output of a simulation model includes screen updates during a simulation run and a report generated after a simulation run.

- System Parameters (Figure 4.7): includes replication index, simulation real-time clock, simulation date, global rock type and Bypass status.
- Ore Output (Figure 4.8): includes run-out time and throughput for each type of ore.
- Blender Output (Figure 4.9): presents blender scheme information for the current and upcoming campaigns.
- Station Output (Figure 4.10): the current station status presents the current inventory level at each station, while the cumulative station status presents the

Output		
Rock Type	Runout	Throughput
Low_Ni	0.0	0.0
Low_Cu	1E+36	1000
Med_Ni	0	0
Med_Cu	1E+36	0
High_Ni	1E+36	0
High_Cu	1E+36	0

Figure 4.8: Ore Output

	Blenders (Current Campaign)					
	Blender A		Blender B		Blender C	
RockType	1	2	3	3	2	3
Percent	0.50	0.50	1.00	0.00	0.30	0.70
Rate	0	0	0	0	0	0
	Blenders (Next Campaign)					
	Blender A		Blender B		Blender C	
RockType	1	2	3	3	2	3
Percent	0.50	0.50	1.00	0.00	0.30	0.70
Rate	0	0	0	0	0	0

Figure 4.9: Blender Output

Current Station Status

Mines	Initial Piles	Primary Crusher	StockPiles	Surge Bin	Crusher	FOBs	GrindMill
0.00	0.00	0.00	0.00	0.00	0.00	0.00	~~0.00~~
0	0.00		0.00			0.00	0.00
~~0~~	0		0			0.00	~~0.00~~
	0		0			0.00	
	0		0			0.00	
	0		0				

Cumulated Station Status

Mines	Initial Piles	Primary Crusher	StockPiles	Surge Bin	Crusher	FOBs	GrindMill
1000.00	562.50	433.33	900.00	1650.00	1250.00	857.50	80.00
1000.000031	333.33		750.00			880.00	0.00
0	0		0			0.00	~~0.00~~
	0		0			0.00	
			0			0.00	
			0				

Figure 4.10: Station Output

cumulative amount of material flow through each station.

- Statistical Results for Throughput (Figure 4.11): presents mean and standard deviations of ore throughput among different replications. Includes real running time to complete the entire simulation run as well.

- Pivot Chart for Current Station Level Data(Figure 4.12): presents a pivot table of current station levels in pivot chart format. The user is able to select "Name of station", "Rock Type" and "Replication Index".

- Pivot Chart of Accumulated Station Level Data(Figure 4.13): presents a pivot table of current station levels in pivot chart format. The user is able to select "Name of station", "Rock Type", and "Replication Index".

- Pie Chart of Maintenance Data(Figure 4.14): presents a pie chart of the percentage of breakdowns of a particular machine. The user is able to select different types of machines.

4.3.4 Running Environment

The model is developed in Microsoft Excel 2007 with Microsoft Visual Basic for Application 6.5. The testing environment is Windows XP Professional with Service

General Information	
Run Time (hrs)	100
Time Step (hrs)	0.25
Replication	5
Run Time (seconds)	18.8

Throughput						Avg	St Dev
Replication	1	2	3	4	5	0	0
Low Ni Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Low Cu Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Med Ni Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Med Cu Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00
High Ni Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00
High Cu Throughput	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 4.11: Statistic Output

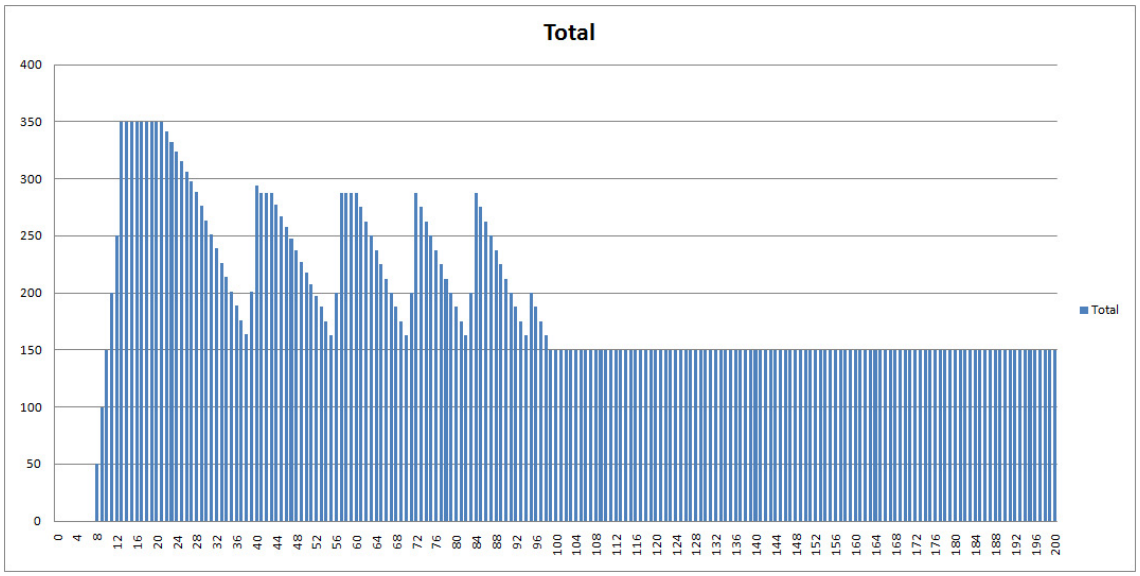

Figure 4.12: Pivot Chart of Current Station Level

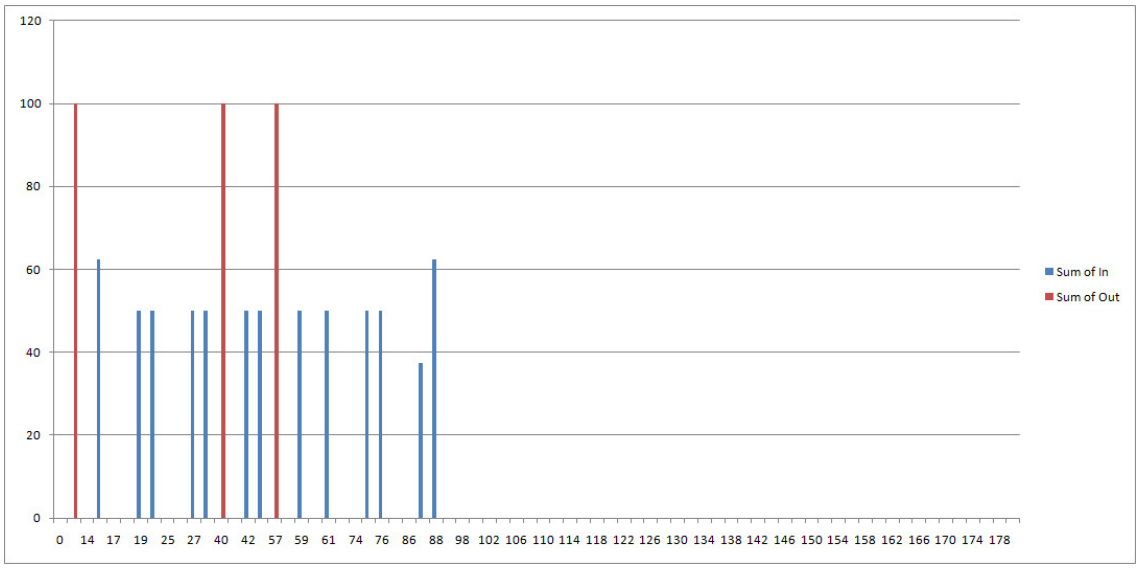

Figure 4.13: Pivot Chart of Accumulated Station Level

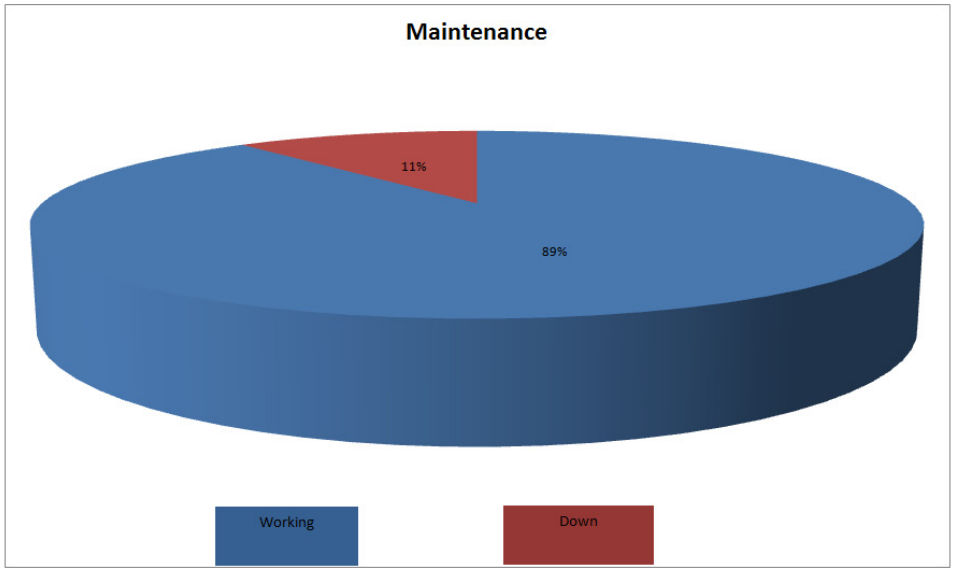

Figure 4.14: Pivot Chart of Station Level

Pack 3. There are several compatibility issues regarding using different versions of software for the simulation model:

i)Pivot chart and pivot table: In Excel 2007, a PivotTable report supports 1,048,576 unique items per field, but in earlier versions of Excel, only 32,500 items per field are supported. In Excel 2007, you can quickly format a PivotTable report by applying a theme-based PivotTable style. In earlier versions of Excel, you can only format a PivotTable report manually. This is one possible reason why users cannot use drop-down menus on pivot charts.

ii).xslm format: Saving the simulation model to .xls (earlier version of Excel 2003) in Excel 2007 increases the file size by 5x. The model is originally programmed in Excel 2007 and then converted to 2003 version. The file size increased dramatically after converting. A possible reason is: the new file type(.xlsm) is a much more compact set of files that are then zipped together. When it is saved back to (.xls), the graphics and features for both versions are put together in the old format.

iii)Windows 7 does not generate the "Save" message window properly when users want to save a .txt output data file.

4.4 Output Analysis

4.4.1 Steady-State Analysis

A system state which is independent of initial starting conditions is defined as *steady-state*. The period the system state response depends on the initial starting condition, defined as a *transient period or warm-up period*. It is necessary to find this period to be able to determine the simulation run length. The simplest and most general technique for determining the warm-up period is the *Welch Method*. Law and Kelton[10] described how Welch's procedure is based on making n independent replications of the simulation and selecting the moving average with proper window size.

For the mill model,the following recommendation was made for choosing the parameters associated with the Welch method analysis:

i)The draw scheme in this steady-state analysis should be selected to be as simple as possible. Thus, it has only one type of ore in the entire grinding circuit, one FOB, and grinding circuit A without any blending option.

ii) *The average current inventory level of a grinding mill* is selected as a performance measure. The simulation model records the inventory level of the grinding mill every 15 minutes, which is the time step of the model. The average inventory level is then calculated correspondingly.

iii)Select n = 5 replications.

iv)select m = 30 days. Since the project is interested in a planning horizon of three months, 30 days is a larger than anticipated value for a warm-up period and also large enough to allow infrequent events to occur a reasonable number of times.

v)Try several values of window w (w = 3, 5 and 7) and choose the smallest value of w for which the corresponding plot is "reasonably smooth".

The sample data collected for the average current level of grinding mill A is presented in Table 4.1, as follows:

Table 4.1: Sample Data

Time step (15 minutes)	**Average**	**moving average w=3**	**moving average w=5**	**moving average w=7**
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.25	0.25
...	...	...	...	...
10	4.00	3.72	3.98	4.24
...	...	...	...	...
15	9.17	8.93	8.63	8.14
...	...	...	...	...
20	11.75	11.67	11.61	11.50
...	...	...	...	...
30	15.18	15.17	15.14	15.10
...	...	...	...	...
40	17.41	17.37	17.34	17.30
...	...	...	...	...
50	18.64	18.63	18.62	18.60
...	...	...	...	...
100	21.42	21.41	21.41	21.40
...	...	...	...	...
200	22.56	22.55	22.55	22.54
...	...	...	...	...
250	22.66	22.66	22.65	22.65
...	...	...	...	...
288	22.66	22.66	22.66	22.66

In Figure 4.15 we plot the average current level of grinding mill A with different window sizes of w = 3, 5 and 7. It was found that the selection of window size does not significantly affect the smoothness of the graphic, and therefore its influence is omitted. In conclusion, the warm-up period is selected as 200 time steps, which is $200 * 15$ minutes = 50 hours.

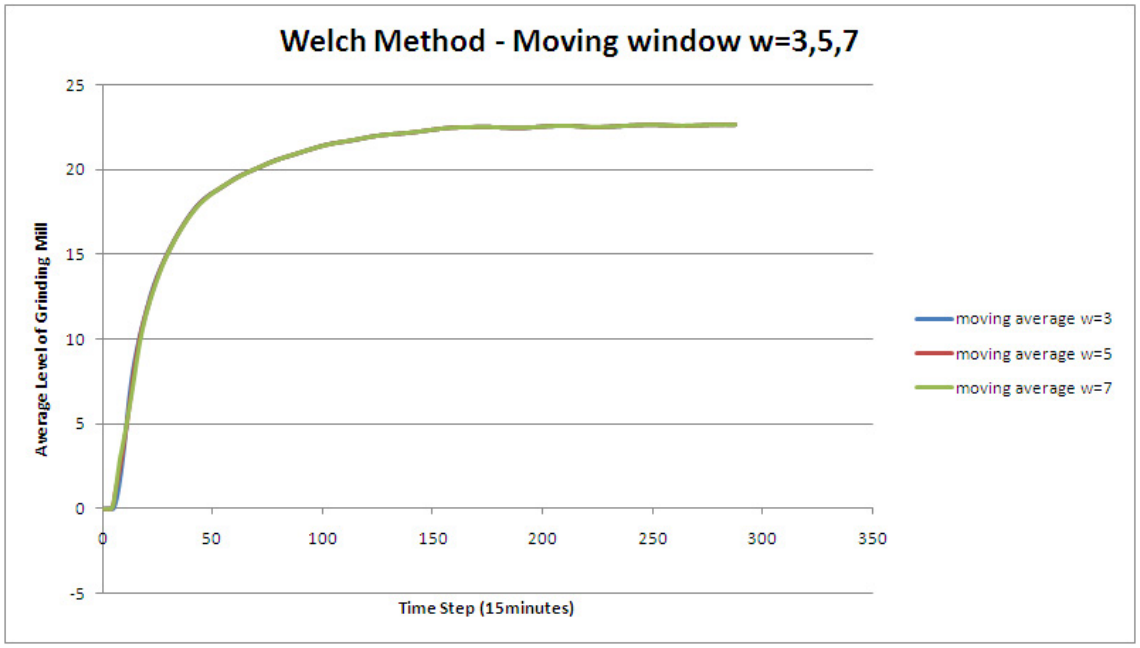

Figure 4.15: Welch Method

4.4.2 Replication Length and Number of Replications

The company does most of its performance analysis on a monthly basis, which includes production forecasting, maintenance scheduling, etc. On the other hand, within a one-month simulation run, it is reasonable that several campaign runs and several machine failures may occur. It was arbitrarily decided the simulation replication length at one month.

However, it is necessary to run the simulation model for enough replications to achieve a desired level of accuracy. There are two approaches to determine the number of replications required: *Absolute Error* and *Relative Error*.

The relative error method is used in this analysis. Law and Kelton[10] give an approximate expression for the number of replications, $n_r^*(\gamma)$, required to obtain a relative error of γ :

$$n_r^*(\gamma) = min\{i \geq n : \frac{t_{i-1,1-\alpha/2}\sqrt{S^2(n)/i}}{|X(n)|} \leq \frac{\gamma}{1-\gamma}\}$$

The simulation setting for this analysis is the same as for the previous one, except for a change in the replication length to 770 hours and in the warm-up period to 50

hours. The simulation model was initially run for five replications. The estimated mean and variance showed as follows:

$$\bar{X}(5) = 21.5526, Var(5) = 1.4027$$

in order to have a relative error of no more than 5% with a confidence interval of 90%. From the five available replications, it was determined that

$$n_r^*(0.1) = min\{i \geq 5 : \frac{t_{i-1,0.95}\sqrt{1.4027/i}}{|21.5526|} \leq \frac{0.05}{0.95}\}$$

$$= min\{i \geq 5 : t_{i-1,0.95}\sqrt{1.4027/i}\} \leq 1.1343$$

Table 4.2 shows results when trying different values of i:

Table 4.2: Sample Data

i	Degree of freedom	t	CI half
2	1	6.31	5.288
3	2	2.92	1.997
4	3	2.35	1.393
5	*4*	*2.13*	*1.129*
6	5	2.02	0.974
7	6	1.94	0.870
8	7	1.90	0.794
9	8	1.86	0.734
10	9	1.83	0.687
11	10	1.81	0.647
12	11	1.80	0.614
13	12	1.78	0.585
14	13	1.77	0.561
15	14	1.76	0.539

Note that the total number of replications actually required is 5, which is same as the initial number of replications tried, thus no additional replications are required.

In the future analysis, the simulation model is run by using replication/deletion approach. This approach was basically the suggested technique for getting n independent runs of the simulation by running the simulator n times with different initial random seeds. In each replication, an appropriately selected warmup period is removed before the data collection is started. For the observed intervals after the warm

up period, data is collected and processed for point estimates of the variables being observed. The replication/deletion method is simple and easy to use.

In the mill simulation model, the user is able to set warmup period and number of replications to perform replication/deletion approach. Thus, roughly 50 hours are are used for warm up. The model runs for another 720 hours (30 days) in steady-state.

4.4.3 Simulation Model Verification and Validation

One of the greatest difficulties in simulation is verification and validation. Verification means to ensure that a simulation model performs as desired, while validation means to determine whether a simulation model truly and accurately presents the actual system.

The guidelines for verification and validation of the mill simulation model follows some general perspectives, as described by Kelton and Law [10]:

i) "A simulation model should always be developed for a particular set of purposes. Indeed, a model that is valid for one purpose may not be for another." The mill simulation model is interested in evaluating the "Campaign and Draw Scheme" solution from the mill optimization model. In other words, the verification of this model should focus observing whether the mill system performs as desired.

ii) "The measure of performance used to validate a model should include those factors that the decision maker will actually use for evaluating system designs." Since the mill model does not have many built-in data collection methods for statistics during a simulation run, the performance measurement of the model will be based on the throughput of each type of ore.

iii) "The ease or difficulty of the validation process depends on the complexity of the system being modelled and on whether a version of the current exists." The mill model intends to evaluate/predict proposed system behaviours. Verification will based on comparing performance of two campaign runs of the same length with similar potential draw scheme.

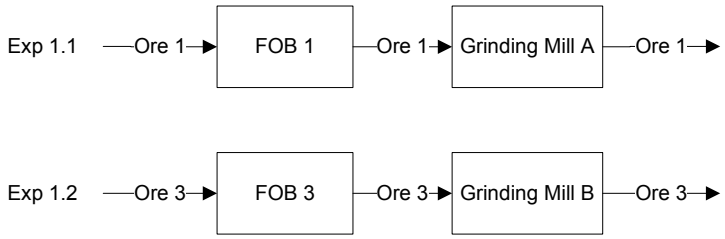

Figure 4.16: Flow Chart of Draw Scheme in Scenario Analysis 1

The following sections demonstrate verification of the simulation model by performing several scenario analysis.

Scenario Analysis 1

One type of ore is processed through one grinding mill which uses one fine ore bin as storage. Draw schemes 1 and 2 are selected in this analysis and their process flow charts are shown in Figure 4.16. The results are shown in Table 4.3.

Table 4.3: Scenario Analysis 1

	Ore Type	FOB	Grinding Mill	Draw Scheme Index	Throughput
Exp 1.1	1	1	A	1	76806.00 ± 65.13
Exp 1.2	3	3	B	2	76815.00 ± 37.91

The throughput results indicate that there are no significant differences between the two configurations. Grinding mill A and B are identical when processing one type of ore using only one FOB.

Scenario Analysis 2

One type of ore is processed through one grinding mill and uses two FOBs as storage at the same time. Draw schemes 3 and 4 are selected in this analysis and their process flow charts are shown in Figure 4.17. The results are shown in Table 4.4.

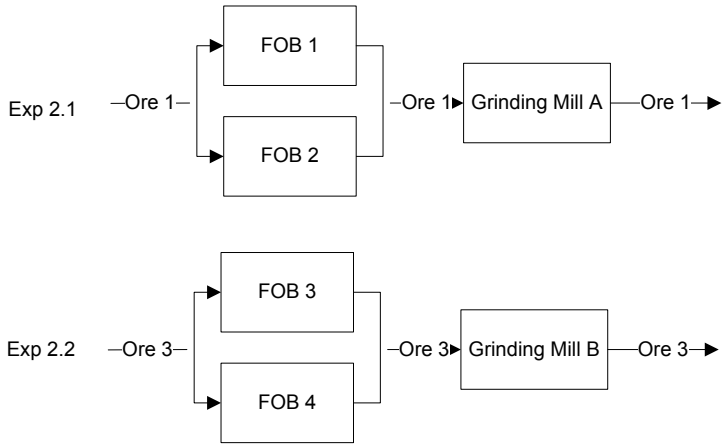

Figure 4.17: Flow Chart of Draw Scheme in Scenario Analysis 2

Table 4.4: Scenario Analysis 2

	Ore Type	FOB	Grinding Mill	Draw Scheme Index	Throughput
Exp 2.1	1	1 and 2	A	3	76796.00 ± 93.10
Exp 2.2	3	3 and 4	B	4	76815.00 ± 22.36

The throughput results indicate that there are no significant differences between two configurations. Grinding mill A and B are identical when processing one type of material using more than one FOB. Comparing results between Exp 1.1 and Exp 2.1, FOB allocation does not matter for one type of ore processed at one grinding mill.

Scenario Analysis 3

Two types of ore are processed through two separate grinding mills and use single FOBs for each type of ore as storage. Draw schemes 5 and 6 are selected in this analysis and their process flow charts are shown in Figure 4.18. The results are shown in Table 4.5.

Table 4.5: Scenario Analysis 3

	Ore Type	FOB	Grinding Mill	Draw Scheme Index	Throughput
Exp 3.1	1	1	A	5	72443.00 ± 150.15
	3	3	B		72093.00 ± 159.48
Exp 3.2	2	2	A	6	72620.00 ± 73.65
	3	3	B		71945.00 ± 87.32

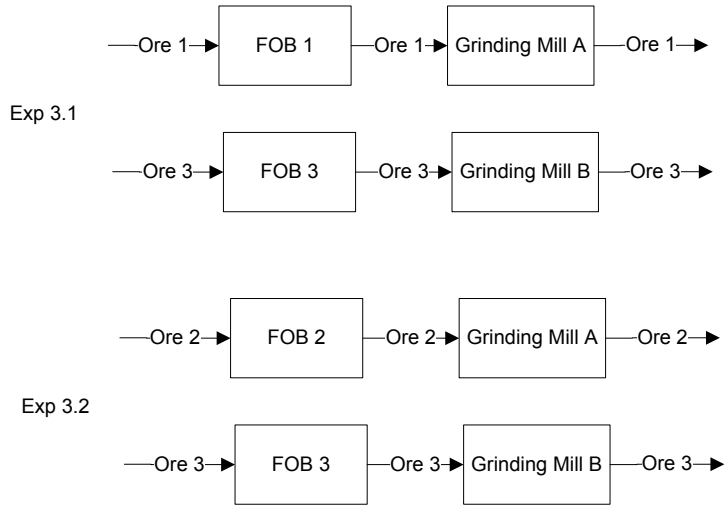

Figure 4.18: Flow Chart of Draw Scheme in Scenario Analysis 3

The throughput results indicate that there are no significant difference between the two configurations. It can thus be concluded that grinding circuit A and B can potentially perform the same under both draw schemes. By comparing results from Exp 1.1, Exp1.2 and Exp3.1, we can see that the throughput is less when there is more than one type of ores being processed at the mill than when compared to one type of ore being processed. This is because crusher and surge bins can only process one type of ore at a time. When another type of ore comes into these two stations, the station has to be emptied first (run to zero inventory). This changeover delay causes a drop in throughput.

Scenario Analysis 4

Two types of ore are processed through one grinding mill with a blending option and use a single FOB for each type of ore as storage. Draw schemes 7 and 8 are selected in this analysis and their process flow charts are shown in Figure 4.19. The results are shown in Table 4.6.

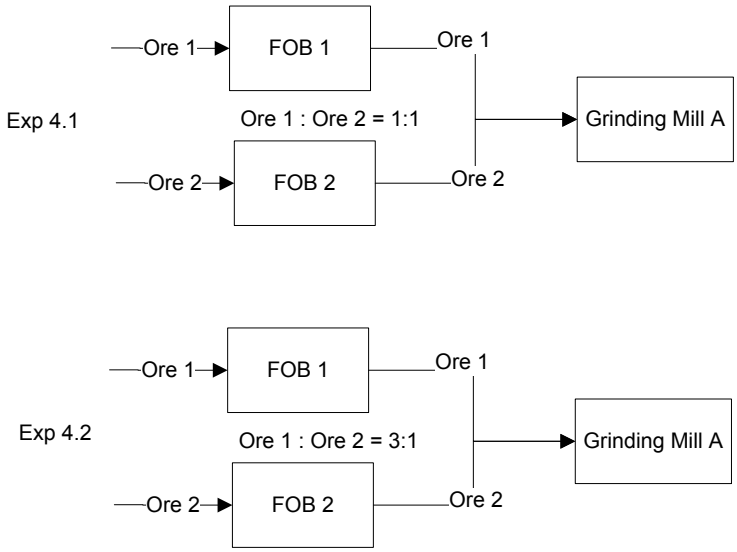

Figure 4.19: Flow Chart of Draw Scheme in Scenario Analysis 4

Table 4.6: Scenario Analysis 4

	Ore Type	FOB	Blending Ratio	Grinding Mill	Draw Scheme Index	Throughput
Exp 4.1	1	1	1:1	A	7	37365 ± 1445.86
	2	2				37365 ± 1445.86
Exp 4.2	1	1	3:1	A	8	56381.25 ± 128.54
	2	2				18793.75 ± 42.85

The total throughput of both draw scheme indicate there are no significant differences between the two configurations. It was found that when the blending option is involved, the grinding circuit's throughput does not decrease significantly. The total throughput from one grinding circuit is slightly less than for the throughput in Exp 1.1.

Scenario Analysis 5

Three types of ore are processed through two grinding mills. Two types of ore are blended at one grinding mill and one type of ore is processed in another grinding mill. Draw scheme 9 is selected in this analysis and its process flow chart is shown in Figure 4.20. The result is shown in Table 4.7.

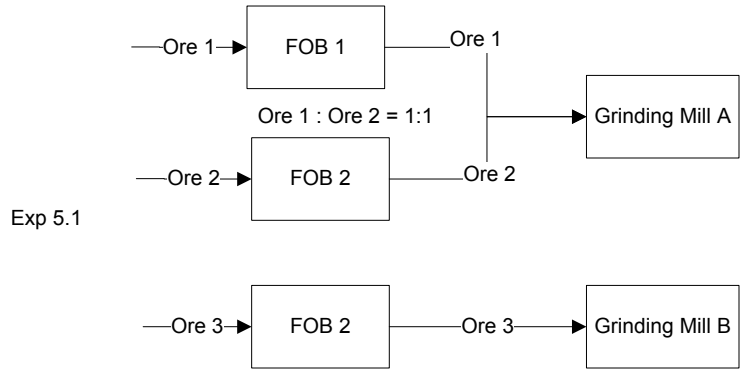

Figure 4.20: Flow Chart of Draw Scheme in Scenario Analysis 5

Table 4.7: Scenario Analysis 5

	Ore Type	FOB	Blending Ratio	Grinding Mill	Draw Scheme Index	Throughput
Exp 5.1	1	1	1:1	A	9	35075.0 ± 142.52
	2	2		A		35075.00 ± 142.52
	3	3		B		64330.00 ± 125.50

Draw scheme 9 likely combines Exp 1.1 and Exp 4.1 in one campaign run. It was observed that both grinding circuit's throughput decreased due to three types of ore being processed in the mill system.

The mill simulation model is verified based on running several campaign runs with similar draw schemes. The potential draw scheme in this verification analysis is created based on the simulation assumption and the complexity level of the problem. The simulation model is used to evaluate a proposed mill system. Once a campaign and a draw scheme concept have been implemented at the mill for a sufficient time, and output data from the system has been collected, the simulation model could be validated by comparing the actual performance with the simulation model's prediction. At that time, the confidence of the "validity" of the model can be increased.

Chapter 5

Machine Failure Data Analysis

Almost all real-world systems contain one or more sources of randomness. In mill operations, machine failure and machine repair are the main sources of randomness. In the simulation model, all activities are performed at a "station", which represents a place where one or many processes are performed in the real situation. Machine failure has dramatic impact on system performance; however, the unpredictability of machine failure is what causes difficulty in machine reliability analysis.

The mill simulation model has the capability of analyzing stochastic elements such as machine failure. It is important to relate physical machines to "stations" in the simulation model so that their failure behaviors can be simulated. The approach for solving this problem is selecting machines contributing the most failures among all machines and trying to categorized them into simulation "stations". These machines' failure data are further fit to the "best" probability distribution as model inputs.

5.1 Selection of Most Frequently Failed Machines

The *80/20 rule* is described as roughly "80% of the effects come from 20% of the causes". In our case, this means it is conceivable that 20% of the total number of machines contribute to 80% of total failures. In this failure analysis, the "80/20" rule was applied to find the equipment that failed most frequently instead of analyzing all equipments.

The failure data are extracted from the company's database source. The data available for this analysis was limited to a one-year data sample ranging from Jan 1st, 2008, to Dec 31st, 2008. In total, the data set recorded 2,123 failure observations from a total of 95 different machines. Figure 5.1 shows the results of the most frequently failed machines by applying the "80/20" rule. According to the above data, 28 types of

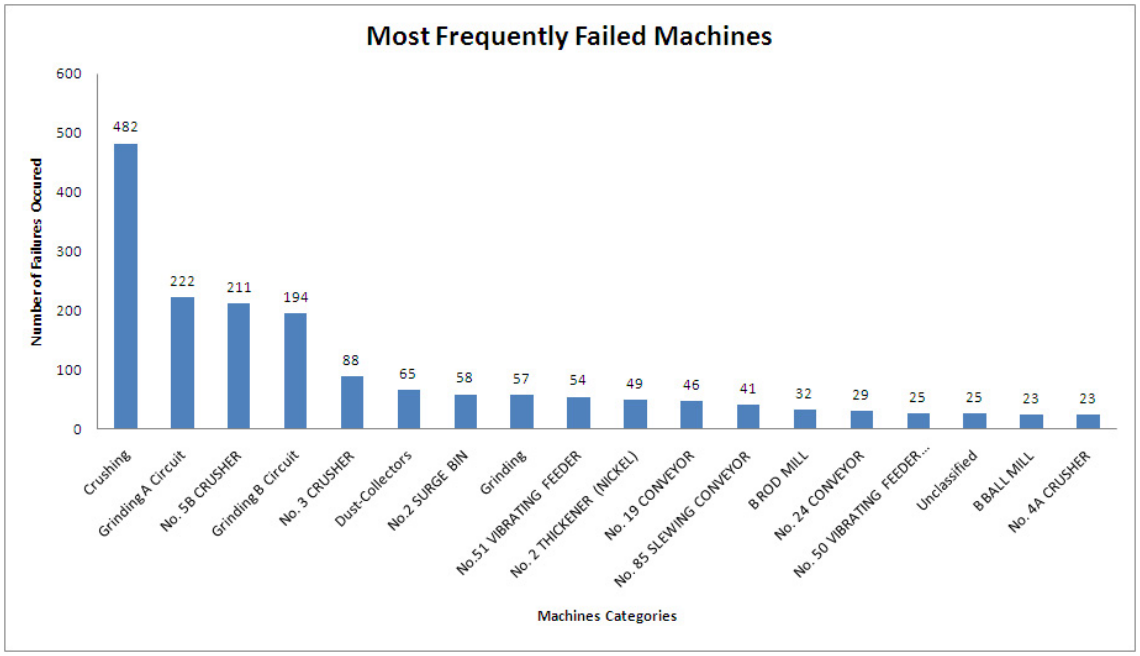

Figure 5.1: Most Frequently Failed Machines

machines contributed 1724/2123 = 81.2% of total failures. The purpose of selecting the most frequently failed machines is to narrow down the data set and focus on critical machines of the mill operation system. From the Figure 5.1, it is found that crushing, grinding circuit, crusher are the three most frequent types of machine failure; and they all had associated "stations" in the simulation model. It is realized that some of the selected machines such as No. 2 Thickener can not be found in a corresponding station in the simulation model. In this case, they are removed from this list. The next step was to explore the relationship between the machines and to simulate machine failure in the model. A block diagram is developed to show the relationship between the selected machines.

These selected machines were then related to the "station" in the simulation model, as follows (Table 5.1):

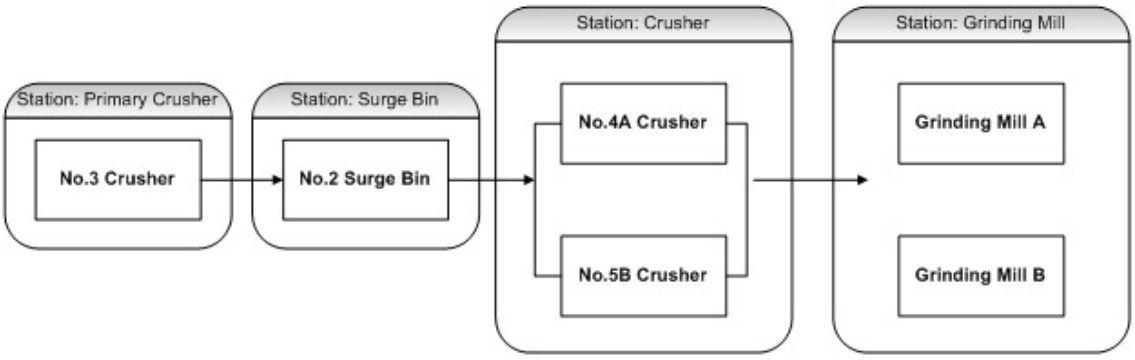

Figure 5.2: Most Frequently Failed Machines

Table 5.1: Stations in Simulation Model

"Station" in Simulation Model	**Equipment Name**
Primary Crusher	No.3 Crusher
Surge Bin	No.2 Surge Bin
Crusher	include No.4A Crusher and No.5B Crusher
Grinder A	Grinding A Circuit
Grinder B	Grinding B Circuit

It was observed that some conveyors are not categorized into any "station". Failure data showed that if the operation is blocked either upstream or downstream (due to, for instance, conveyor breakdown), it is defined as an operating standby failure. Although conveyors do not have a "station" in the simulation model, their failures are still counted as upstream or downstream equipment failure. A final simplified block diagram is shown in Figure 5.2.

5.2 Failure Data Analysis

5.2.1 Types of Failure

There are four types of failure collected in the maintenance data set:

i) *Scheduled Maintenance*: Includes two types of maintenance work. One is periodically repeated maintenance work such as preventive maintenance. Another is one-time scheduled work for major repairs, cleaning, and so on.

This type of maintenance is scheduled and is planned and controlled by maintenance personnel.

ii) *Unscheduled Maintenance*: Unscheduled maintenance is related to a random failure of a machine due to a mechanical problem. This type of failure is unpredictable.

iii) *Operating Problem*: There are operational interruptions caused by external factors which can not be controlled by maintenance groups, such as poor quality of incoming material, lack of trucks, and so on. In this case, the operation has to be interrupted until the problem is solved.

iv) *Operating Standby*: This involves operation interruptions caused by a block originating either from upstream or downstream, such as no feed from the upstream or failure of downstream. In this case, equipment has to wait for a certain amount of time or is to run at low speed.
In the simulation logic, if a "station" is in breakdown mode, its upstream cannot deliver material to this station which causes the upstream station station to be in *standby* status; if a "station" reaches its maximum or minimum inventory level, it cannot receive or deliver ore to its upstream or downstream, this causes the upstream and downstream to be in *standby* statuses. Since the simulation model accommodates operating standby failure as part of its operating logic, this type of failure is excluded from further failure analysis to avoid double-counting the failures. It was found that Grinding A circuit that $139/222 = 62.6\%$ of the failures are operating standby; Grinding B circuit $136/194 = 70.1\%$. However, other machines such as No. 4A and 5B crusher has no operating standby failure and No. 2 surge bin has 3.4% of total failures belong to operating standby failures.

Each failure has its own characteristics and should be treated independently. It is ideal to perform failure analysis for each type of failure for each type of machine. A large amount of data is required to perform this type of analysis. However, based on Figure 5.1, it was found that most equipment fails less than 50 times per year.

If these failures are further divided into the four categories according to four failure types, only a small data sample is available for each type of failure. Since statistical distributions was used as simulation model input, small data samples usually not fit an acceptable distribution.

Assumptions were thus made to overcome this situation:

i)Failure data analysis includes three types of failures: Scheduled Maintenance, Unscheduled Maintenance and Operating Problem.

ii)Since the only interest is in MTTR and MTBF for machines, three types of failures: Scheduled Maintenance, Unscheduled Maintenance and Operating Problem were pooled for failure data analysis purposes.

5.2.2 Mean Time Between Failure and Mean Time to Repair

There are two types of failure data which are used in the simulation model as input: Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR). The software "ExpertFit" was used to perform the distribution fitting process for this work. Distribution fitting is the procedure of selecting a statistical distribution that best fits a data set. In most cases, it fits two or more distributions, compares the results, and selects the most valid model based on Chi-square, Kolmogorov-Smirnov, or Anderson-Darling goodness-of-fit tests.

Mean Time Between Failure

The definition of MTBF depends on the definition of system failure. In this analysis, all four types of failures are considered as system failures. MTBF is the elapsed time between failures of a system during operation. MTBF is therefore mean "UP" time between any two failures.

Within the recorded maintenance data set, when a repair work lasts longer than one shift, the maintenance recording system treats it as two or more consecutive failure records. In this case, all these sequenced failures have to be manually combined into one failure. The distribution fit results are presented in Table 5.2.

Table 5.2: Mean Time Between Failures (in Days)

MTBF Station	Distribution Type	Distribution Parameters			Mean	Standard Deviation
		Location	Scale	Shape		
No.3 Crusher	Weibull	0.79	6.70	0.76	7.29	9.95
No.2 Surge Bin	Weibull	1.18	5.489	0.62	8.77	11.34
No.4A Crusher	Weibull	1.02	10.55	0.60	16.51	22.22
No.5B Crusher	Weibull	1.27	6.59	0.66	9.51	8.93
Grinding Mill A	Gamma	0.00	22.38	0.44	5.03	10.76
Grinding Mill B	Gamma	0.00	13.86	0.66	9.10	9.28

To select proper distribution type for input data samples, a related score which is a number between 0 and 100 that gives an indication of how well a fitted distribution represents a data set. In this case, the distribution type with the highest related score was considered first. If random variable generator for this type distribution was available in the simulation model, it was used. Otherwise, the next type of distribution with highest score was considered next. The results indicated that most MTBF follows either a Weibull or Gamma distribution. Scale, location and shape, as distribution parameters, are used as inputs for the simulation model.

Mean Time to Repair

MTTR measures the average time to repair a failure. In this analysis, due to lack of detailed information, the MTTR may include lead time for parts not available, maintenance group response time, and other downtime. The definition of MTTR is the elapsed time starting from equipment breakdown until it is fixed and back to operational status.

Similarly, when a major failure requires more than one shift to repair, the repair is recorded as several separate repair records. In this case, repair time should be summed. The distribution fit results are presented in Table 5.3.

Table 5.3: Mean Time to Repair (in Minutes)

MTTR Station	**Distribution** Type	**Distribution Parameters**			**Mean**	**Standard Deviation**
		Location	**Scale**	**Shape**		
No.3 Crusher	Weibull	22.00	248.41	0.45	758.83	2330.76
No.2 Surge Bin	Lognormal	0.00	243.47	0.84	343.45	304.54
No.4A Crusher	Weibull	16.00	58.58	0.42	174.37	348.89
No.5B Crusher	Weibull	23.97	1084.11	0.51	1965.90	3245.16
Grinding Mill A	Weibull	106.11	1698.31	0.70	2253.60	2927.88
Grinding Mill B	Weibull	0.00	932.32	0.684	1,255.71	2378.54

5.3 Failure Consequence-Partial Failure

5.3.1 Definition of Partial Failure

When machine experiences partial failure, the throughput levels during this period will be less than 100%. *Failure Efficiency* is defined, as follows, to represent this phenomenon:

$$FailureEfficiency = 1 - (\frac{EffectiveDowntime}{FailureStart - FailureEnd})$$

The transfer rate of any station should be calculated as follows:

$$TransferRate = RegularTransferRate \times FailureEfficiency$$

When a station works without a failure, the transfer rate is 100% of its regular transfer rate; otherwise, the transfer rate is reduced. Figure 5.3 demonstrates the way failure efficiency is applied in the simulation model.

5.3.2 Failure Efficiency Implementation

To implement the failure efficiency concept in the model, the first step was to find the percentage of time when the failure has zero throughput (p indicated in Figure 5.3). The result of this step is presented in Table 5.4.

The next step was to find the "best" distribution to generate failure efficiency "x", which is indicated in Figure 5.3 and Table 5.5. Results are not available for "No.4A Crusher" due to the lack of data.

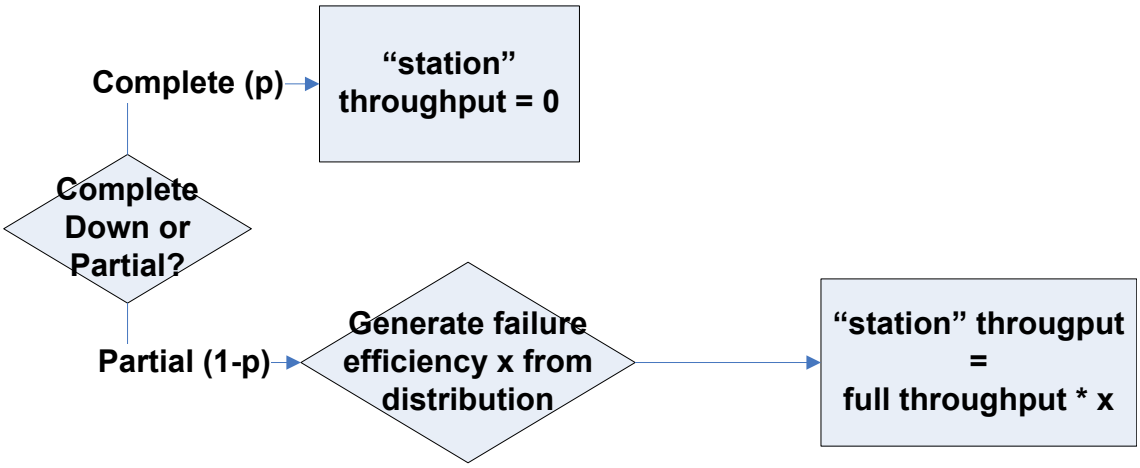

Figure 5.3: Simulation logic for applying failure efficiency

Table 5.4: Percentage of Time Machine Completely Broke

	No.3 Crusher	No.2 Surge Bin	No.4A Crusher	No.5B Crusher	Mill A	Mill B
Total Down	88	58	23	211	222	194
Completely Down	82	35	21	102	183	151
Partially Down	6	23	2	109	39	43
% of completely Down	93.18%	60.34%	91.30%	48.34%	82.43%	77.84%
% of Partial Down	6.82%	39.66%	8.70%	51.66%	17.57%	22.16%

Table 5.5: Failure Efficiency Distribution

	Distribution Type	Distribution Parameters		
Station		**Location**	**Scale**	**Shape**
No.3 Crusher	Gamma	0	0.0127	55.386
No.2 Surge Bin	Gamma	0	0.013	50.970
No.4A Crusher	-	-	-	-
No.5B Crusher	Weibull	0	0.723	7.129
Grinding Mill A	Weibull	0.3739	0.4879	4.8549
Grinding Mill B	Gamma	0	0	130.9159

Chapter 6

Simulation - Based Sensitivity Analysis for Examining Alternatives

In general, the optimization model focuses on finding an optimal solution that minimizes or maximizes a linear objective function from millions of possible alternatives, while meeting given constraints. It simplifies the representation of real world problems. This simulation model was found to be able to handle a highly complex system which contains nonlinear and stochastic elements. One feature of simulation is that the parameters of a simulation model can be changed easily to observe system performance under different sets of parameters. Therefore, simulation-optimization can be defined as a method to ascertain a set of parameters that determine optimal solution for the system performance.

A simulation model usually has controllable and uncontrollable factors. In the mill simulation model, the campaign schedule and its draw scheme are considered as uncontrollable factors since they are believed to be the best alternatives found through the IP model. Then simulation-based sensitivity analysis of the mill finds the combination of controllable factors that optimizes the simulation response. Since the draw scheme includes many fixed control policies for the mill, the simulation is used as an evaluation tool rather than an optimization tool.

In this section, simulation-based sensitivity analysis is used to look at a few controllable factors. At the same time, the interest is in single factor response. The analysis is scenario-based optimization and is described as follows:

i)Select the *Base Model* (6.1) for this analysis. Throughput is the performance measure.

ii)Experiment One: Sensitivity analysis of MTTR and MTBF.

iii)Experiment Two: Sensitivity analysis of increased arrivals.

6.1 Base Model

It is assumed that the base model has four campaigns and no machine failures. The process flow charts are shown in Figure 6.1. Results of base model are compared with other scenarios and shown in Table 6.2.

6.2 Sensitivity Analysis of MTTR and MTBF

Scenario Analysis 1: Maintenance Implemented Model

The second scenario analysis include MTTR and MTBF for machines. At this point, only grinding circuit A experiences failures. The input distribution in this sensitivity analysis is selected as a uniform distribution. The uniform distribution is good estimation when only information available about the a parameter X is that it between limits a and b. In this case, MTTR and MTBF is the mean value of the uniform distribution. The input distribution are shown in Table 6.1

Scenario Analysis 2: Increased Mean Time Between Failure

It is assumed that the MTBF can be increased by roughly 10% which means better machine availability is achieved. In this case, MTBF from scenario one is increased by 10% The new input of this analysis are shown in Table 6.1.

Scenario Analysis 3: Reduced Mean Time to Repair

It is assumed that the MTTR can be reduced by roughly 10% which means better machine availability is achieved. In this case, MTTR from scenario one is reduced by 10%. The new inputs of this analysis are shown in Table 6.1.

The overall results of the sensitivity analysis of MTTR and MTBF are shown in Table 6.2. By comparing all three scenarios' result to the base model, we can see

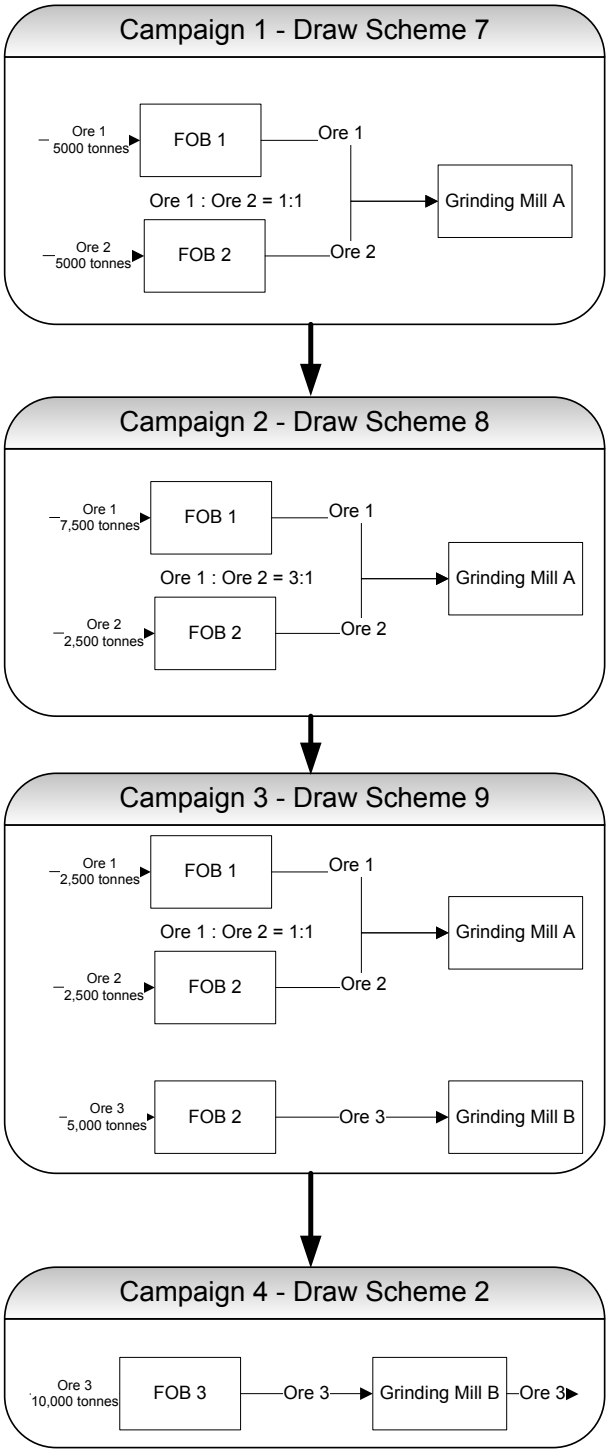

Figure 6.1: Process flow chart of base model

Table 6.1: Input Parameters of Uniform Distribution for MTTR and MTBF Sensitivity Analysis

	MTBF			MTTR		
	Para 1 (a)	Para 2 (b)	Mean	Para 1 (a)	Para 2 (b)	Mean
Scenario 1	90	100	95	10	30	20
Scenario 2	100	110	105	10	30	20
Scenario 3	90	100	95	9	27	18

that the throughput fluctuates more when machines experience failure. Note, larger MTBF means machines experience less failures during their life time. During the four campaign runs, it was observed that only few failures occured. This indicated that there are no significant differences between all three scenarios and the base model. Comparing scenario 2 and 3 to the scenario 1, fluctuations in throughput are observed. However, improved machine availability in Scenario 2 and 3 did not resulted in significantly different throughput.

The sensitivity analysis of MTTR and MTBF presented here is a basic guide for future analysis. Although scenario-based optimization may not include all alternatives, it does provide a method for a decision maker to use the simulation model as a tool for conducting similar sensitivity analysis. Future analysis could include applying MTTR and MTBF to all machines; using more accurate MTTR and MTBF from a larger data sample; and machine availability and reliability analysis.

6.3 Sensitivity Analysis of Increased Arrival Ore

Evaluating the sensitivity of increased arrivals is one way to test the system's capacity. It is assumed that arrival of each type of ore from each mine is increased by 10%. The result is compared to the base model and shown in Table 6.3. It is observed that the output of each type of ore from the base model and the scenario 4 model is less than the input of ore. In other words, the mill system is "saturated" due to its limited capacity. It was also realized that increased ore arrival may change the "sequence" with which ores are processed. This is the reason of output of ore type 3's dropped from 11000 in base model to 7700 in scenario 4 model.

Table 6.2: Results of Selectivity Analysis of MTTR and MTBF

		Total Input	Total Output	
		Mean	Mean	St Dev
Base Model	Ore Type 1	15000	11306	0
	Ore Type 2	10000	7069	0
	Ore Type 3	15000	11000	0
Scenario 1	Ore Type 1	15000	13467.5	1452.5
	Ore Type 2	10000	8092.5	2024.7
	Ore Type 3	15000	7042.0	2836.0
Scenario 2	Ore Type 1	15000	14606.5	2247.2
	Ore Type 2	10000	7562.5	236.3
	Ore Type 3	15000	5845.0	2570.5
Scenario 3	Ore Type 1	15000	12880.1	2146.9
	Ore Type 2	10000	7662.0	1110.8
	Ore Type 3	15000	7155.0	1496.1

Table 6.3: Scenario Analysis: Increased arrival of Ore

		Input of Ore	Output of Ore	Total Throughput
Base Model	Ore Type 1	15000	11306	
	Ore Type 2	10000	7069	
	Ore Type 3	15000	11000	29375
Scenario 4	Ore Type 1	16500	14606.2	
	Ore Type 2	11000	8518.8	
	Ore Type 3	16500	7700.0	30825

Chapter 7

Conclusion and Future Studies

In this research, simulation and mathematical programming are combined for optimization purposes. The simulation runs at a small fraction of real-time, and the mathematical programming guides the search for good solutions without the need to evaluate all possibilities. The approach can be briefly described as followed: the mill integer programming model evaluates and selects the "best" solution subject to certain constraints. The outcome of the model is a series of campaigns with specific draw scheme that minimize the excess inventory levels of unprocessed ore and the number of days on shutdown and the number of active piles required at any point in time. The simulation model then takes this solution as its input and evaluate system performance and perform certain simulation - based sensitivity analysis.

As a first useful result, the champaign and draw schemes problem has been formulated as an integer programming problem implemented in GLPK language using GUSEK. A small set of numerical experiments has been carried out. The potential draw schemes and other required data were made up to solve the problem. The sample problem is to define 4 campaigns during a 20 day planning horizon, and to choose a draw scheme from 4 potential draw schemes that includes one shutdown draw scheme. The results demonstrated that the IP model is able to solve the relatively simple problem within a reasonable execution time. The proposed IP model can be expanded in the future to include the selection of more complicated draw schemes; the model should be tested with real data.

The simulation model was constructed by following the "station" concept. A "pull system" concept ensures a continuous material flow simulation model can be represented by a discrete-event simulation model. The completed mill simulation model includes features such as campaign and draw scheme; integrated random failure

and scheduled maintenance; integrated mine production delivery; multiple types of ore and customer-demand driven approach. The verification analysis of the simulation model tested ten potential draw schemes that ranged from a scenario which processed one type of ore to processing three types of ore with a blending option. The model was verified by comparing the throughput of the mill against different draw schemes. The output analysis of the simulation model included determination of the length of warm-up period and number of replications. The proposed simulation model was used to perform simulation - based sensitivity analysis. The analysis measures changes in throughput by improving the equipment uptime ratio and increased arrival rate of ore. The next step for simulation model might be to include more complicated draw schemes, which involve more types of ores in the system, and running changeovers between campaigns.

Lastly, as a part of input data analysis for simulation model, maintenance data was analyzed. The data was first filtered by selecting the most frequently failed machines as analysis targets. The MTTR and MTBF for each machine was then analyzed according to different types of failures. Statistical software was used to find the best distribution. The results indicated that most MTTR follows the Weibull distribution and MTBF follows either the Weibull or Gamma distribution. The maintenance data analysis gives a basic idea of how to interpret failure data and relate it to reliability engineering concepts. A continuing analysis and follow-up when further data become available is recommended.

Overall, these two models can be used not only by the Canadian mining company, but also for others. The mill optimization model can be a general production planning tool for any mine-mill operation. The model is flexible enough to have different types of ores, arrival rates from mine production, and inventory capacity limits. The objective function of this model gives the decision maker the ability to emphasize on inventory control, shutdown planning and/or stockpiles usage. This can be implemented by assigning proper weight factors to these three goals in the objective function. Besides the mill optimization model, the blending model can be used individually with consideration of ore grade along with draw scheme development. Secondly, the mill simulation model provides a basic framework for modelling

mill ore process. Draw schemes and campaigns are the two essential elements in the model. The user is able to create their own draw scheme and simulate the ore processing using the simulation and thus evaluate the throughput. The use of a draw scheme provides an easy way to change the routine of ore processing. It can be treated as an input to the simulation model which thus enhances the flexibility of the simulation model. On the other hand, a campaign can be defined either as one long campaign or series of campaigns depend on the decision maker's interest. One long campaign can be used to investigate influence of fluctuating factors on the mill throughput. However, series of campaigns gives an idea of the impact of transitions between campaigns to those interested in campaign changeover.

In summary, the achievements obtained from this research work include:

i) Production planning is implemented by the introduction of the campaign concept: a consecutive number of days which use the same draw scheme, to configure and control the mill production system considering throughput as the performance measures.

ii) A novel simulation model, implemented in VBA using Excel as the interface has been developed. This simulation uses a very flexible object oriented model based on the concept of station. Stations transfer material according to certain control rules. The model is essentially a continuous time model where the user has the option to specify the calculation time step. The concept is flexible enough that it is possible to simulate a variety of mill configurations.

iii) The proposed production plan generated from the IP model is evaluated by the simulation model to include random machine failures and changeovers. When failure are included, the IP model could be used to generate a new feasible production plan based on feedback from the simulation model.

Bibliography

[1]Agioutantis Z.G., & Stratakis A.(1998). Simulation of a Continuous Surface Mining System Using the Micro Saint Visual Simulation Package, In *Information technology in the mineral industry*. Panagiotou, G.N. & T.N. Michalakopoulos (eds.). Rotterdam: Balkema, p.85 and CD-ROM.

[2]Chinbat, U., & Takakuwa, S. (2008), Using Operation Process Simulation for a Six Sigma Project of Mining and Iron Production Factory. Paper presented at the WSC'08: Proceedings of the 40th Conference on Winter Simulation, Miami, Florida. 2431-2438.

[3]Erdem, B., Celebi, N., & Pasamehmetoglu, A.G. (1997). A Computer Simulation Model for Gragline Stripping in Surface Coal Mines with One Flat-lying Seam. In *Mine Simulation.* Panagiotou, G.N. & J.R. Sturgul (eds.), Rotterdam: Balkema, p.3 and CD-ROM.

[4]Fioroni, M. M., Franzese, L. A. G., Bianchi, T. J., Ezawa, L., Pinto, L. R., & de Miranda, J.,Gilberto. (2008). Concurrent Simulation and Optimization Models for Mining Planning. Paper presented at the WSC '08: Proceedings of the 40th Conference on Winter Simulation, Miami, Florida. 759-767.

[5]GLPK (GNU Linear Programming Kit). Retrieved Sep 20, 2010, from http://www.gnu.org/software/glpk/.

[6]GNU Operating System. Retrieved Sep 20, 2010, from http://www.gnu.org/.

[7]Gunn,E.A. (2009)Collaborative research and development (CRD) grants progress report. 6-7.

[8]GUSEK (GLPK Under Scite Extended Kit). Retrieved Sep 20, 2010, from http://gusek.sourceforge.net/gusek.html.

[9]Huband, S.,While, L., Tuppurainen, D., Hingston, P., Barone, L. & Bearman, T. (2006) Economic Optimisation of an Ore Processing Plant with a Constrained Multi-objective Evolutionary Algorithm. In A.Satter and B.H. Kang (Eds),AI 2006: Advances in Artificial Intelligence, Vol 4304 (pp. 962-969)

[10]Kelton, W. D., & Law, A. M. (1982). In Simulation Modelling and Analysis. McGraw-Hill series in industrial engineering and management science. New York: McGraw-Hill. (8, 78, 243-244, 501-502, 509)

[11]Kutcha, M., Newman, A., & Topal, E. (2003). Production Scheduling at LKAB's Kiruna Mine Using Mixed Integer Programming.

[12]Kuchta, M., Newman, A., & Topal, E. (2004). Implementing a Production Schedule at LKAB's Kiruna Mine.

[13]Lynch, A.J. & Morrison, R.D. (1999) Simulation In Mineral Processing History, Present Status And Possibilities. Western Cape Branch Conference: Mineral Processing '99 Aug. 1999.

[14]Medved, B. & Runovc, F., (1997). Computer Analysis of Truck Transport in Uranium Mine. In *Mine Simulation.* Panagiotou, G.N. & J.R. Sturgul (eds.), Rotterdam: Balkema, p.102 and CD-ROM.

[15]Mutagwaba, W. & Durucan, S. (1993) . Object-oriented Simulation in Mine Transportation Design. In *Mine Mechanization and Automation.* Almgren, G., U. Kumar, & N. Vagenas (eds.), Rotterdam: Balkema, pp.591-600.

[16]Panagiotou, G.N. & Michalakopoulos, T.N., (1997). STRAPAC2: A Tool for Planning and Analysis of Shovel-Truck Operations. In *Mine simulation.* Panagiotou, G.N. & J.R. Sturgul (eds.), Rotterdam: Balkema, p.7 and CD-ROM.

[17]Panagiotou, G. N. (1999). Discrete Mine System Simulation in Europe.13(2), 43.

[18]SciTe: a free cource code editor for Win32 and X. Retrieved Sep 20, 2010, from http://www.scintilla.org/SciTE.html.

[19]Sturgul, J. R., & Li, Z. (1997). New Developments in Simulation Technology and Applications in the Minerals Industry.11(4), 159.

[20]Svedensten, P., & Evertsson, C. M. (2005). Crushing Plant Optimisation by Means of a Genetic Evolutionary Algorithm. Minerals Engineering, 18(5), 473-479.

[21]Turner, R. J. (1999). Simulation in the Mining Industry of South Africa. International Journal of Mining, Reclamation and Environment, Volume 13, Issue 2 1999, 47 - 56.

[22]Vagenas, N. & Forsman, B., (1992). METAFORA: A Simulator for Dispatch Control of Truck/shovel Systems in Surface Mines. Simulation Conf. and Annual Meeting of the Scandinavian Simulation Society (SIMS 92), June 10-12, Lapeenranta, Finland.

[23]Vagenas, N. (1999). Applications of Discrete-event Simulation in Canadian Mining Operations in the Nineties.13(2), 77.

[24]Venter, J.J., Bearman, R.A., & Everson, R.C.(1997). A Noval Approach to Circuit Synthesis in Mineral Processing. Mineral Engineering 10(3), 99.287-299.

[25]Wilke, F. L. (1970) : Simulation Studies of Computer Controlled Traffic Underground in Large Coal Mines, 9th Int. Symp. on Decision Making in the Mineral Industry, Can IMM, sp. vol.12, pp 344-351.

[26]Wilke, F.L. & Keck, K. (1982). Simulation Studies of Truck Dispatching, *17th APCOM Processings*, Col. School of Mines, Golden, CO and pub. by SME, Littleton, CO, pp 620-626

[27]While, L., Barone, L., Hingston, P., Tuppurainen, D., & Bearman, R., (2004). A Multi-objective Evolutionary Algorithm Approach for Crusher Optimization and Flowsheet Design. Mineral Engineering 17 (11/12), pp. 287-299.

Appendices

Appendix A

LP of Mill Optimization Model - GUSEK Code

```
/* Mill Optimization Model*/
/* Written in GNU MathProg*/

/*Define*/
/*1. Campaign*/
param nDays,integer;/*Number of days of planning horizon*/
param nDrawSchemes,integer;/*Number of potential draw schemes*/
param CLmax,integer;/*Maximum campaign length in DAYS*/
param CLmin,integer;/*Minimum campaign length in DAYS*/
param N,integer;/*Number of Campaigns required*/

set Days, default {1..nDays};/*Set of days*/
set Arcs := { (i,j) in {Days cross Days}: i <= nDays-CLmin and j>= CLmin and (j-i)>= CLmin and (j-i)<=CLmax};
set DrawSchemes, default {0..nDrawSchemes};/*Draw schemes set, 0 means shutdown*/
var Zsd,integer;/*The number of days on shutdown during the planning horizon*/
var x{(i,j) in Arcs},binary;/*x[ij] = 1 means campaign starts at the beginning
of day i and ends at the beginning of day j*/
var y{(i,j) in Arcs, k in DrawSchemes},binary;/* y[ijk] =1 means draw scheme k
is used during campaign i to j*/

/* 2. Material flow and pils limit*/
param nOres,integer;/*number of different types ore*/
set Ores, default {1..nOres};/*Set of Ores*/

param Itot_max{t in Days};/*Maximum total inventory allowed during period t*/
param Itot_min{t in Days};/*Minimum total inventory allowed during period t*/
param Iore_max{o in Ores, t in Days};/*Maximum inventory allowed for ore type o during period t*/
param Iore_min{o in Ores, t in Days};/*Minimum inventory allowed for ore type o during period t*/
param a{o in Ores, t in Days}, >=0;/*Arrival of ore type O in period t*/
param D{o in Ores, k in DrawSchemes}, >=0;/*Draw rate per day for ore type O under draw scheme k*/

var d{o in Ores, t in Days},>=0;/*draw of ore type O in period t*/
var inv{o in Ores, t in Days},>=0;/* Ending inventory of ore type O at the end period t*/
var Vtot_upper{t in Days},>=0;/*Violation on upper of total ending inventory limit during period t*/
var Vtot_lower{t in Days},>=0;/*Violation on lower of total ending inventory limit during period t*/
var Vore_upper{o in Ores, t in Days},>=0;/*Violation on upper of total ending
inventory limit for rock type o during period t*/
var Vore_lower{o in Ores, t in Days},>=0;/*Violation on lower of total ending
inventory limit for rock type o during period t*/

/*3. Active piles*/
var z{ o in Ores, t in Days},binary;/*z[ot] = 0 means ore type o pile active within period t*/
var Vnp;/*variable of maximun number of active piles allowed in any periods*/
param NPmax;/*maximun number of active piles allowed in any periods*/
param Gamma;
param Beta;


/*Objective function*/
```

```
minimize obj:  sum{t in Days} Vtot_upper[t] + sum{t in Days}Vtot_lower[t] +
sum{o in Ores,t in Days}Vore_upper[o,t] + sum{o in Ores,t in Days}
Vore_lower[o,t]  - Gamma*Zsd + Beta*Vnp ;

/*Constrants*/
/*1. Campaign*/
s.t. CampaignRequired: sum{(i,j) in Arcs} x[i,j] = N;/*Total campaign required for palnning horzon*/
s.t. Equal{b in 2..nDays-1: b > 2}: sum{(m,b) in Arcs} x[m,b] = sum{(b,n) in Arcs} b[k,n];/* b>2 is enforcement*/
s.t. StartDay: sum {j in Days: (1,j) in Arcs} x[1,j] =1;
/*Make sure the campaign start from day 1*/
s.t. OneDrawScheme{(i,j) in Arcs}:sum {k in DrawSchemes} y[i,j,k] = x[i,j];
/*Choose one draw scheme in each campaign period*/
s.t. Shutdown: Zsd = sum{(i,j) in Arcs, k in DrawSchemes: k=0} y[i,j,k] * (j-i);
/*Confirm total # of shutdown days*/

/*2. Material flow and pile limits*/
s.t. Draw{o in Ores, t in Days}: d[o,t] = sum {(i,j) in Arcs, k in DrawSchemes: i<=t and j>t} D[o,k]*y[i,j,k] ;
/*Daily draw equals pre-calculated draw rate/day if draw schem k used*/
s.t. Inventory{o in Ores, t in Days:t>1}: inv[o,t]=inv[o,t-1]+a[o,t]-d[o,t];
/*Ending Inv = Ending Inv of previous day +Arrival - Draw*/

s.t. TotalInvLimit{t in Days}: Itot_min[t] <= sum{o in Ores} inv[o,t]-
Vtot_upper[t] + Vtot_lower[t] <=Itot_max[t];
/*At any end of givin day, total inventory should btw lower and upper bound*/
s.t. OreInvLimit{o in Ores, t in Days}: Iore_min[o,t] <= inv[o,t] -
Vore_upper[o,t] + Vore_lower[o,t] <= Iore_max[o,t];
/*At any end of givin day, for each type of ore, inventory should btw lower and upper bound*/

/*3. Active piles*/
s.t. PileInv{o in Ores,t in Days}: inv[o,t] <= z[o,t]* sum{p in 1..t} a[o,p];
s.t. PileDraw {o in Ores, t in Days}: d[o,t]<= z[o,t]* max{k in DrawSchemes} D[o,k] ;
s.t. NumActive{t in Days}: sum{o in Ores} z[o,t] <= NPmax + Vnp;/*Total number of active piles*/
s.t. Numbershutdown:Zsd >=1;

solve;

display {(i,j) in Arcs, k in DrawSchemes: y[i,j,k]=1} y[i,j,k];

end;
```

Appendix B

LP of Mill Optimization Model - External Data File

```
data;

param nDays:= 20;
param nDrawSchemes:=3;
param CLmax:=7;
param CLmin:=2;
param N:=4;
param nOres:=3;
param NPmax:=5;
param Gamma:=24;
param Beta:=1.2;

param a:=
1 1 80
1 2 80
1 3 80
1 4 80
1 5 80
1 6 70
1 7 70
1 8 70
1 9 70
1 10 70
1 11 70
1 12 70
1 13 70
1 14 70
1 15 70
1 16 70
1 17 70
1 18 80
1 19 80
1 20 90
2 1 70
2 2 70
2 3 70
2 4 70
2 5 70
2 6 80
2 7 80
2 8 80
2 9 80
2 10 80
2 11 70
2 12 70
2 13 80
2 14 90
2 15 80
2 16 90
```

```
2 17 70
2 18 70
2 19 70
2 20 70
3 1 90
3 2 90
3 3 60
3 4 80
3 5 80
3 6 70
3 7 60
3 8 80
3 9 80
3 10 70
3 11 70
3 12 70
3 13 70
3 14 70
3 15 70
3 16 65
3 17 65
3 18 65
3 19 65
3 20 65

;
/*D{o in Ores, k in DrawSchemes}*/
param D:=
/*draw scheme 0:shutdown */
1 0 0
2 0 0
3 0 0

/*draw scheme 1 */
1 1 40
2 1 50
3 1 30

/*draw scheme 2*/
1 2 50
2 2 30
3 2 40

/*draw scheme 3 */
1 3 30
2 3 40
3 3 50

;

param Itot_min:=
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
```

```
14 0
15 0
16 0
17 0
18 0
19 0
20 0

;

param Itot_max:=
1 1000
2 1000
3 1000
4 1000
5 1000
6 1000
7 1000
8 1000
9 1000
10 1000
11 1000
12 1000
13 1000
14 1000
15 1000
16 1000
17 1000
18 1000
19 1000
20 1000

;

param Iore_max:=
1 1 300
1 2 300
1 3 300
1 4 300
1 5 300
1 6 300
1 7 300
1 8 300
1 9 300
1 10 300
1 11 300
1 12 300
1 13 300
1 14 300
1 15 300
1 16 300
1 17 300
1 18 300
1 19 300
1 20 300
2 1 300
2 2 300
2 3 300
2 4 300
2 5 300
2 6 300
2 7 300
2 8 300
2 9 300
```

```
2 10 300
2 11 300
2 12 300
2 13 300
2 14 300
2 15 300
2 16 300
2 17 300
2 18 300
2 19 300
2 20 300
3 1 300
3 2 300
3 3 300
3 4 300
3 5 300
3 6 300
3 7 300
3 8 300
3 9 300
3 10 300
3 11 300
3 12 300
3 13 300
3 14 300
3 15 300
3 16 300
3 17 300
3 18 300
3 19 300
3 20 300

;

param lore_min:=
1 1 0
1 2 0
1 3 0
1 4 0
1 5 0
1 6 0
1 7 0
1 8 0
1 9 0
1 10 0
1 11 0
1 12 0
1 13 0
1 14 0
1 15 0
1 16 0
1 17 0
1 18 0
1 19 0
1 20 0
2 1 0
2 2 0
2 3 0
2 4 0
2 5 0
2 6 0
2 7 0
2 8 0
2 9 0
```

```
2  10  0
2  11  0
2  12  0
2  13  0
2  14  0
2  15  0
2  16  0
2  17  0
2  18  0
2  19  0
2  20  0
3  1  0
3  2  0
3  3  0
3  4  0
3  5  0
3  6  0
3  7  0
3  8  0
3  9  0
3  10  0
3  11  0
3  12  0
3  13  0
3  14  0
3  15  0
3  16  0
3  17  0
3  18  0
3  19  0
3  20  0

;

end;
```

Appendix C

LP of Mill Optimization Model - Output File

```
GLPSOL: GLPK LP/MIP Solver 4.38
Reading model section from Mill_v5.mod...
81 lines were read
Reading data section from Mill_v5.dat...
276 lines were read
Generating obj...
Generating CampaignRequired...
Generating Equal...
Generating StartDay...
Generating OneDrawScheme...
Generating Shutdown...
Generating Draw...
Generating Inventory...
Generating TotalInvLimit...
Generating OreInvLimit...
Generating PileInv...
Generating PileDraw...
Generating NumActive...
Generating Numbershutdown...
Model has been successfully generated
ipp_basic_tech:  15 row(s) and 33 column(s) removed
ipp_reduce_bnds: 4 pass(es) made, 258 bound(s) reduced
ipp_basic_tech:  131 row(s) and 58 column(s) removed
ipp_reduce_coef: 1 pass(es) made, 0 coefficient(s) reduced
glp_intopt: presolved MIP has 306 rows, 716 columns, 4655 non-zeros
glp_intopt: 439 integer columns, 438 of which are binary
Scaling...
 A: min|aij| = 1.000e+000  max|aij| = 9.000e+001  ratio = 9.000e+001
GM: min|aij| = 6.174e-001  max|aij| = 1.620e+000  ratio = 2.623e+000
EQ: min|aij| = 3.934e-001  max|aij| = 1.000e+000  ratio = 2.542e+000
2N: min|aij| = 2.500e-001  max|aij| = 1.094e+000  ratio = 4.375e+000
Crashing...
Size of triangular part = 305
Solving LP relaxation...
      0: obj = -2.368840000e+004  infeas = 6.094e+004 (1)
    200: obj =  2.591960000e+004  infeas = 2.041e+003 (1)
    400: obj =  2.779960000e+004  infeas = 5.000e-001 (1)
*   404: obj =  2.778960000e+004  infeas = 1.945e-013 (1)
*   562: obj =  3.009600000e+003  infeas = 5.094e-013 (1)
OPTIMAL SOLUTION FOUND
Integer optimization begins...
Gomory's cuts enabled
MIR cuts enabled
Cover cuts enabled
Clique cuts enabled
Creating the conflict graph...
The conflict graph has 2*435 vertices and 114261 edges
+   562: mip =     not found yet >=              -inf        (1; 0)
```

```
+  4365: >>>>>  1.713360000e+004 >=  1.058360000e+004  38.2% (9; 0)
+  6057: >>>>>  1.195960000e+004 >=  1.067667317e+004  10.7% (9; 2)
+  8091: >>>>>  1.191960000e+004 >=  1.187960000e+004   0.3% (4; 14)
+ 14583: mip =  1.191960000e+004 >=  1.188667726e+004   0.3% (33; 51)
+ 15714: mip =  1.191960000e+004 >=     tree is empty   0.0% (0; 187)
INTEGER OPTIMAL SOLUTION FOUND
Time used:    10.2 secs
Memory used: 3.4 Mb (3512890 bytes)
Display statement at line 79
y[1,7,1] = 1
y[7,13,3] = 1
y[13,18,1] = 1
y[18,20,0] = 1
Model has been successfully processed
Writing MIP solution to 'Mill_v5.out'...
>Exit code: 0    Time: 10.402
```

Appendix D

LP of Blending Model - GUSEK Code

```
param nOre;
param nBlender;
set Ore,default {1..nOre};/*Set of Ores*/
set Blender,default {1..nBlender};/*Set of Blenders*/

param CuGrade{o in Ore};/*Ore's copper grade*/
param NiGrade{o in Ore};/*Ore's Nickel grade*/

param FOBRatio;
param MinNi;/*Same for MinCu */
param penalty;

var x{o in Ore,b in Blender},>=0;
var y{o in Ore, b in Blender},binary;
var z{b in Blender},>=0;
var CuContent,>=0;
var NiContent,>=0;
var del1,>=0;
var del2,>=0;

minimize obj:(del1+del2)*penalty + sum{o in Ore, b in Blender} y[o,b];

s.t. OreBinUsed{o in Ore,b in Blender}: x[o,b] <= y [o,b];
s.t. Bin{b in Blender}: sum{o in Ore} x[o,b] = z[b];
s.t. OneTon: sum{b in Blender} z[b] =1;
s.t. CopperContent: CuContent = sum{o in Ore, b in Blender} x[o,b]*CuGrade[o];
s.t. NickelContent: NiContent = sum{o in Ore, b in Blender} x[o,b]*NiGrade[o];
s.t. EqualMetal: NiContent <= CuContent + del1;
s.t. EqualMetal_1:CuContent <= NiContent + del1;
s.t. BlendingRatio{b in Blender: b>=1 and b<nBlender}: z[b] <= FOBRatio*z[b+1];
s.t. BlendingRatio_1{b in Blender:b>=1 and b<nBlender}: z[b+1] <= FOBRatio*z[b];
s.t. MinimumNi:NiContent + del2 >= MinNi;
s.t. MinimumCu:CuContent + del2 >= MinNi;

solve;

data;

param nOre:=3;
param nBlender:=2;
param CuGrade:=
1 6.1
2 2.6
3 1.4;

param NiGrade:=
1 1.7
2 2.2
3 2.6;

param FOBRatio:= 2;
param MinNi:=2.8;
```

```
param penalty:=1000;

end;
```

Appendix E

LP of Blending Model - Solution Output File

```
Problem:     Blend
Rows:        18
Columns:     18 (6 integer, 6 binary)
Non-zeros:   58
Status:      INTEGER OPTIMAL
Objective:   obj = 395.8571429 (MINimum)
```

No.	Row name		Activity	Lower bound	Upper bound
1	obj		395.857		
2	OreBinUsed[1,1]		-0.785714		-0
3	OreBinUsed[1,2]		0		-0
4	OreBinUsed[2,1]		0		-0
5	OreBinUsed[2,2]		0		-0
6	OreBinUsed[3,1]		-0.547619		-0
7	OreBinUsed[3,2]		-0.666667		-0
8	Bin[1]		0	-0	=
9	Bin[2]		0	-0	=
10	OneTon		1	1	=
11	CopperContent		0	-0	=
12	NickelContent		0	-0	=
13	EqualMetal		0		-0
14	EqualMetal_1		0		-0
15	BlendingRatio[1]		0		-0
16	BlendingRatio_1[1]		-1		-0
17	MinimumNi		2.8	2.8	
18	MinimumCu		2.8	2.8	

No.	Column name		Activity	Lower bound	Upper bound
1	x[1,1]		0.214286	0	
2	x[1,2]		0	0	
3	x[2,1]		0	0	
4	x[2,2]		0	0	
5	x[3,1]		0.452381	0	
6	x[3,2]		0.333333	0	
7	y[1,1]	*	1	0	1

```
     8 y[1,2]       *              0            0            1
     9 y[2,1]       *              0            0            1
    10 y[2,2]       *              0            0            1
    11 y[3,1]       *              1            0            1
    12 y[3,2]       *              1            0            1
    13 z[1]                 0.666667            0
    14 z[2]                 0.333333            0
    15 CuContent             2.40714            0
    16 NiContent             2.40714            0
    17 del1                        0            0
    18 del2                 0.392857            0

Integer feasibility conditions:

KKT.PE: max.abs.err = 2.44e-014 on row 1
        max.rel.err = 3.70e-017 on row 10
        High quality

KKT.PB: max.abs.err = 1.97e-016 on row 12
        max.rel.err = 1.97e-016 on row 12
        High quality

End of output
```

Appendix F

Standard Module Code

Subroutine RunSim is the starting point of the simulation. The first part of RunSim is initialization. It clears the output content on the Excel output sheet; sets the simulation clock; and asks user to save the output file; The next step uses a loop to go through different replication (For i = 1 To intReps): at the beginning of each loop, several sub routines runs first to perform initialization to downtime, schedule etc. Then subroutine simRummer is called to run one replication of the simulation. At the end of each replication loop, statistic are calculated and displayed on the output sheet.

```
Public Sub RunSim()
    %Dim intReps As Integer
    Dim i As Integer
    Dim clsResults() As RepResult

    Dim objDialog As Object
    Dim intResult As Integer

    Dim iFNumber As Integer
    Dim lrow As Double

    intReps = Range("rngNumReps")
    ReDim clsResults(-2 To intReps)

    Set clsResults(-1) = New RepResult      %Will store sum of results
    Set clsResults(-2) = New RepResult      %Will store (sum of results)^2

    %Nov 2009
    Sheets("stats_StnLevel").Range("A:ZZ").ClearContents

    %Pop up Windows File Dialog Window to save txt file
    MsgBox "Save_result_file_as_.txt_file"
    Set objDialog = CreateObject("UserAccounts.CommonDialog")
    objDialog.Filter = "Text_Files|*.txt|All_Files|*.*"
    objDialog.FilterIndex = 1
    objDialog.InitialDir = "C:"
    intResult = objDialog.ShowOpen

    If intResult = 0 Then
        Exit Sub
    End If
```

```
sFName = objDialog.FileName

%Get File Directory with File Name - Nov 2009

Set collResultFiles = Nothing
Dim currentFileName As ResultFile
    Set currentFileName = New ResultFile
    currentFileName.GetFileName = sFName
    collResultFiles.Add currentFileName

%MsgBox currentFileName.GetFileName


%If output.txt exists, delete it first
If Len(Dir$(sFName)) > 0 Then
    MsgBox "File " & currentFileName.GetFileName & " Exists. About to Delete"
    Kill sFName
End If

%Set Simulation Start Time
sngStart = Timer

For i = 1 To intReps

    %Get a new result class instance
    Set clsResults(i) = New RepResult

    %Output to screen
    Application.StatusBar = "Replication " & CStr(i) & " of " & CStr(intReps)
    Range("rngReplication") = i

    %Clear the collections
    ClearCollection collStnList
    ClearCollection collBlender
    ClearCollection collDowntime

    %Read campaign information
    Call initializeCampaign

    %Initialize variables
    Call initialize

    %Initialize the downtime elements
    Call InitDowntime

    %Initialize mines schedule
    Call InitMinesSchedule

    %Initialize the schedule elements
    Call InitSchedules

    %Run a replication of the simulation
    Call SimRunner(clsResults(i))

    %Update the replication number
    clsResults(i).intRepNumber = i


    %Sum of squares
    clsResults(-2).dblLowNiOutput = clsResults(-2).dblLowNiOutput + clsResults(i).dblLowNiOutput ^ 2
    clsResults(-2).dblLowCuOutput = clsResults(-2).dblLowCuOutput + clsResults(i).dblLowCuOutput ^ 2
    clsResults(-2).dblMedNiOutput = clsResults(-2).dblMedNiOutput + clsResults(i).dblMedNiOutput ^ 2
    clsResults(-2).dblMedCuOutput = clsResults(-2).dblMedCuOutput + clsResults(i).dblMedCuOutput ^ 2
    clsResults(-2).dblHighNiOutput = clsResults(-2).dblHighNiOutput + clsResults(i).dblHighNiOutput ^ 2
```

```
        clsResults(-2).dblHighCuOutput = clsResults(-2).dblHighCuOutput + clsResults(i).dblHighCuOutput ^ 2

        %Sum of values
        clsResults(-1).dblLowNiOutput = clsResults(-1).dblLowNiOutput + clsResults(i).dblLowNiOutput
        clsResults(-1).dblLowCuOutput = clsResults(-1).dblLowCuOutput + clsResults(i).dblLowCuOutput
        clsResults(-1).dblMedNiOutput = clsResults(-1).dblMedNiOutput + clsResults(i).dblMedNiOutput
        clsResults(-1).dblMedCuOutput = clsResults(-1).dblMedCuOutput + clsResults(i).dblMedCuOutput
        clsResults(-1).dblHighNiOutput = clsResults(-1).dblHighNiOutput + clsResults(i).dblHighNiOutput
        clsResults(-1).dblHighCuOutput = clsResults(-1).dblHighCuOutput + clsResults(i).dblHighCuOutput


    Next i


    %Get the mean and stdev
    %Sheets("Model").Select
    If intReps > 0 Then

        Range("rngLowNiOutput").Offset(0, 0) = clsResults(-1).
        dblLowNiOutput / intReps
        Range("rngLowNiOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblLowNiOutput, clsResults(-2).
        dblLowNiOutput, intReps)
        Range("rngLowCuOutput").Offset(0, 0) = clsResults(-1).
        dblLowCuOutput / intReps
        Range("rngLowCuOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblLowCuOutput, clsResults(-2).
        dblLowCuOutput, intReps)

        Range("rngMedNiOutput").Offset(0, 0) = clsResults(-1).
        dblMedNiOutput / intReps
        Range("rngMedNiOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblMedNiOutput, clsResults(-2).
        dblMedNiOutput, intReps)
        Range("rngMedCuOutput").Offset(0, 0) = clsResults(-1).
        dblMedCuOutput / intReps
        Range("rngMedCuOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblMedCuOutput, clsResults(-2).
        dblMedCuOutput, intReps)

        Range("rngHighNiOutput").Offset(0, 0) = clsResults(-1).
        dblHighNiOutput / intReps
        Range("rngHighNiOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblHighNiOutput, clsResults(-2).
        dblHighNiOutput, intReps)
        Range("rngHighCuOutput").Offset(0, 0) = clsResults(-1).
        dblHighCuOutput / intReps
        Range("rngHighCuOutput").Offset(0, 1) = GetSTD(clsResults(-1).dblHighCuOutput, clsResults(-2).
        dblHighCuOutput, intReps)

    End If

        %Collect Stats- Nov 2009
        Call Stats

    Application.StatusBar = ""

    %Set model end time
    sngEnd = Timer
    Sheets("stats_Summary").Range("c7").Value = Format(sngEnd - sngStart, "Fixed")
End Sub
```

Appendix G

Sample Code of Class Module - Blender

1. Part A: Declare the properties for Blender Class Module. These values will be stored in private variables within the class which they cannot be accessed outside the class module.
2. Part B: Declare Property procedures to allow these variables to be read from and written to. This is done with Property Get and Property Let functions. The "Get" procedure is used to return a value out of the class, and the "Let" procedure is to put a value into the class. A property can be made read-only simply by omitting the Let procedure, such as "Number" property.
3. Part C: Function in class module behaves as a method of the object like any public function procedure. A function is a method that can generate a return value. Sample function code "setPercent" calculate blending ratio between rock type A and B. It reads value from properties and return the value to "blenderPercentA" and "blenderPercentB".
4. Part D: Set up collection in a class module gives user control over interaction with the collection, and the code is encapsulated into a single module that makes it more transportable and easier to maintain. The standard collection's methods (Add,Count,Item, and Remove) cannot be use in the class module. Sample code "RemoveStation" and "AddStation" demonstrated the own method for Add and Remove methods in the class module.
5. Part E: Creating blender collection in a standard module. At the top of the standard module, *collBlender* is declared to be a new collection and *blnCurrent* is declared as a new blender class object. After properties are assigned to *blnCurrent* class module, it is added to the *collBlender* and then set to *nothing*. Do while ... Loop is used to add all blender to the *collBlender*.

```
**************************Part A*********************************************
This class object is a blender - it sits between the FOBs and the Grind mills
These are the properties of a blender

Private blenderName As String                Name
Private blenderNumber As Integer             Number
Private blenderMill As Station               The actual grind mill

Private collFOB As New Collection            A collection to keep track of the
FOBs assigned to the mill
Private blenderOutputRange As Range      The output range (assume nickel)

ROCK TYPE SPECIFIC VARIABLES
Private blenderRockTypeA As Integer          The type of rock A is.
Private blenderPercentA As Single            The percent of rock type A being mixed
Private collFOBRockTypeA As New Collection   A collection to keep track of the Rock Type A FOBs
Private blenderRunOutA As Single             The time to run out of Rock Type A
Private blenderRateA As Single               The rate of consumption of Rock Type A
Private blenderAmountA As Single             The amount of Rock Type A to transfer

Private blenderRockTypeB As Integer          The type of rock B is.
Private blenderPercentB As Single            The percent of rock type B being mixed



Private collFOBRockTypeB As New Collection   A collection to keep track of the Rock Type B FOBs
Private blenderRunOutB As Single             The time to run out of Rock Type B
Private blenderRateB As Single               The rate of consumption of Rock Type B
Private blenderAmountB As Single            The ammount of Rock Type B to transfer

**************************Part B*********************************************
%Gets index of rocktypeA for blender
Public Property Get RockTypeA() As Integer
    RockTypeA = blenderRockTypeA
End Property

%Gets index of rocktypeB for blender
Public Property Get rockTypeB() As Integer
    rockTypeB = blenderRockTypeB
End Property

%These functions will need developing as we introduce new rock types
Public Property Let RockTypeA(intRockType As Integer)
    blenderRockTypeA = intRockType
End Property

Public Property Let rockTypeB(intRockType As Integer)
    blenderRockTypeB = intRockType
End Property

%This gets the name
Public Property Get Name() As String
    Name = blenderName
End Property

%This property sets the name
Public Property Let Name(strBlenderName As String)
    blenderName = strBlenderName
End Property

%This gets the number
Public Property Get Number() As Integer
    Number = blenderNumber
```

```
End Property

%This property sets the number
Public Property Let Number(intNumber As Integer)
    blenderNumber = CStr(intNumber)
End Property

%This property gets the grind mill for this blender
Public Property Get Mill() As Station
    Set Mill = blenderMill
End Property

%This property sets the grind mill for this blender
Public Property Let Mill(stnMill As Station)
    Set blenderMill = stnMill
End Property

%This method adds an FOB to the FOB collection
Public Sub AddFOB(stnFOB As Station)
    collFOB.Add stnFOB
End Sub

%Gets the output range for the blender
Public Property Let OutputRange(rngRange As Range)
    Set blenderOutputRange = rngRange
End Property

%Sets the output range for the blender
Public Property Get OutputRange() As Range
    Set OutputRange = blenderOutputRange
End Property


**************************Part C*********************************************
Function setPercent(intRockType As Integer, sngPercent As Single)

    %Check to see if rocktype A is the mentioned rock type
    If blenderRockTypeA = intRockType Then

        %Set the percent for Rock Type A, then 100% - [this %] for B
        blenderPercentA = sngPercent
        blenderPercentB = 1 - sngPercent

    ElseIf blenderRockTypeB = intRockType Then

        %Set the percent for Rock Type B, then 100% - [this %] for A
        blenderPercentB = sngPercent
        blenderPercentA = 1 - sngPercent

    %If this is not one of the rock types being blended we have an error
    End If

End Function

**************************Part D*********************************************
%This method removes a station from a collection.  It is internal to the
%class object
Private Sub RemoveStation(collCollection As Collection, stnLocal As Station)

    Dim i As Integer
    Dim stnCurrent As New Station

    For i = 1 To collCollection.count
        Set stnCurrent = collCollection(i)
```

```
        If stnCurrent.Name = stnLocal.Name Then
            collCollection.Remove (i)
            Exit Sub
        End If
    Next i
End Sub

%This method adds a station to a collection in decreasing order of
%currentlevel.  It is internal to the class object
Private Sub AddStation(collCollection As Collection, stnLocal As Station)
    Dim i As Integer
    Dim intStart As Integer
    Dim stnCurrent As New Station

    intStart = 1

    If collCollection.count > 0 Then
        Set stnCurrent = collCollection(1)

        %If the current station is on the top of the list and has product, continue to draw
        %from it & exit the sub
        If stnCurrent.Name = stnLocal.Name And stnCurrent.CurrentLevel > stnCurrent.MinLevel Then
            Exit Sub

        %If the station on the top of the list has product, continue to draw from it and
        %put the new station onto the list starting at the 2nd station
        ElseIf stnCurrent.CurrentLevel > stnCurrent.MinLevel Then
            intStart = 2
        End If

        %Remove the station from the list to keep from having duplicates
        Call RemoveStation(collCollection, stnLocal)

    End If


    For i = intStart To collCollection.count
        Set stnCurrent = collCollection(i)
        If stnCurrent.CurrentLevel < stnLocal.CurrentLevel Then
            collCollection.Add stnLocal, , i
            Exit For
        End If

    Next i

    If i > collCollection.count Then
        collCollection.Add stnLocal
    End If

End Sub


**************************Part E*********************************************
Public Sub InitializeBlender()
    Dim collBlender As New Collection
    Dim i, j As Integer
    Dim rngCopy As Range
    Dim rngDestination As Range
    Dim intIndex As Integer

    Dim blnLocalList() As Blender
    Dim blnCurrent As New Blender

    %set collBlender
```

```
Set collBlender = Nothing
Set collNextBlender = Nothing

%Loop through the blender input parameter sheet and store the values in
%Blender Class Module properties.
intIndex = 1
Do While Range("rngBlenderDef").Offset(0, intIndex) <> ""

    ReDim Preserve blnLocalList(intIndex)

    blnCurrent.Name = Range("rngBlenderDef").Offset(0, intIndex)
    blnCurrent.Number = Range("rngBlenderDef").Offset(1, intIndex)
    blnCurrent.OutputRange = Range(Range("rngBlenderDef").Offset(2, intIndex))

    blnCurrent.RockTypeA = Range("rngBlenderDef").Offset(4, intIndex)
    blnCurrent.rockTypeB = Range("rngBlenderDef").Offset(5, intIndex)
    Call blnCurrent.setPercent(blnCurrent.RockTypeA,
    Range("rngBlenderDef").Offset(3, intIndex))
    blnCurrent.Mill = GetStation(Range("rngBlenderDef").Offset(6, intIndex))
    j = 1
    Do While Range("rngBlenderDef").Offset(j + 6, intIndex) <> "" And j <= 10
        blnCurrent.AddFOB GetStation(Range("rngBlenderDef").Offset(j + 6, intIndex))
        j = j + 1
    Loop

    Set blnLocalList(intIndex) = blnCurrent
    Call AddBlenderToCollection(blnCurrent, collBlender)
    Set blnCurrent = Nothing

intIndex = intIndex + 1
Loop
```