Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1992

A Simulation Game for Software Project Management

Ibrahim Bayraktutar
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-002

A Simulation Game for Software Project Management

Ibrahim Bayraktutar

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

A Simulation Game for Software Project Management

by

Ibrahim Bayraktutar

Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-002 March 1992

A SIMULATION GAME FOR

SOFTWARE PROJECT MANAGEMENT

Graduate Research

by

Ibrahim Bayraktutar

In Partial Fulfillment
of the Requirements for the Degree
Master in Systems Analysis

Miami University
February, 1992

TABLE OF CONTENTS Page #

Approvals

Student: _____ Date: 3-3-92

Advisor: _____ Date: 3-3-92

Member: _____ Date: 3-3-92

Member: _____ Date: 3-6-92

ACKNOWLEDGMENT

I would like to convey my appreciation to my advisor Dr. Yaman Barlas for his valuable guidance and support throughout the research. I am grateful to Dr. James Kiper and Dr. Don Byrkett, the other members of the Advisory Committee, for their helpful criticisms and suggestions.

ABSTRACT

The software industry is one of the most rapidly growing businesses in our age. Yet, this growth has not been a well-balanced one. On the one hand, on the technical side, programming languages, programming techniques and methodologies have exhibited an unprecedented growth (1). But on the other hand, the art and science of managing software projects has not enjoyed such a growth. Management of software systems has been plagued by overruns, late deliveries, and users' dissatisfaction (1).

The objective of this project is to design an interactive simulation game based on a system dynamics model of software project management. It is hoped that the game will provide insights into the effectiveness of various management strategies, especially when the project is behind schedule.

The project comprised three major steps. The first step was to construct a dynamic model of how software projects are managed in software developing organizations. After studying several models that already exist in the literature (4,5), I decided to simplify and modify the system dynamics model (in Dynamo) constructed by Tarek-Abdel Hamid and Stuart Madnick (1). In order to construct a model (in Stella II) that can be the background of a participatory simulation game, I made certain modifications, additions and deletions. I simulated this model under various conditions and studied its properties in order to make sure that it was a reliable and robust model.

The second step of the project was to complete the gaming interface for the developed model. For this purpose, several well-established principles in constructing informative and user-

friendly computer gaming interfaces were utilized (2,6,7). The final game interface is graphical and consists of two major environments (the "simulator controls" and the "information system"), and numerous pop-up windows. In order to implement the game in an IBM-PC environment, the game was coded using the graphics-based spreadsheet software WingZ.

The third and final step was testing and validating the system. The system dynamics model that I modified for the game had already been subjected to validity testing and was found to be quite reliable (1). In addition, I tested my version of the model by running it under various typical and extreme conditions. Tests of the gaming version have demonstrated the game to be robust even under various extreme and unlikely player decisions. Tests with independent player subjects have also confirmed this and have demonstrated the game to be potentially very useful. The overall behavior of the game satisfies my goal of constructing an interactive simulation game which is able to reflect the characteristics of real life software management projects, and respond to player decisions in a realistic way so as to provide an interactive learning laboratory for software project management.

I. INTRODUCTION

Our increasingly interconnected and dynamic world challenges managers to find new ways to understand and control change. Software industry is one of the most rapidly growing business in our age. As one can easily anticipate, this growth has not been painless. The software management literature indicates that the development of software systems has been plagued by overruns, late deliveries, and users' dissatisfaction (1).

A major deficiency in large scale software project management is the inability to integrate our knowledge of the individual components of the software development process to derive implications about the behavior of the total system. For this purpose, there is a growing trend in combining system dynamics models with computer-based case studies in order to create realistic models (1,2,3,4). Such studies promote improvement in strategic thinking skills and better integration of isolated operational decisions in the policy and strategy area.

Recently, an exciting new approach to understanding complex dynamic problems has emerged: Interactive simulation gaming (2,5,6). In this approach, one builds an interactive simulation model of some dynamic problem of interest, which allows the user to participate in and influence the course of a given simulation. Interactive simulation games motivate learning, create a situation within which players can experience a wide variety of complex phenomena that have been previously unfamiliar to them, convey principles of system behavior, enhance the group's skills in communication and decision making, evaluate specific decisions and provide a context for system research and evaluation (2,5).

With an interactive simulation game, the computer can be used efficiently to explore a large number of meaningful experiments and search for winning strategies.

The objective of this project is to design an interactive simulation game based on a system dynamics model of software project management problem. It is hoped that the game will provide insights into the effectiveness of various management strategies, especially when the project is behind schedule. The system includes both management-type functions as well as software production-type activities. An important feature that the interactive environment brings is the use of feedback principles of system dynamics to structure and clarify the complex web of dynamically interacting variables.

The project comprises three major steps: The first is to construct a dynamic model of how software projects are actually managed in software developing organizations. The dynamic model of the game is constructed by synthesizing the models that already exist in literature (4,5) and by utilizing methodology and principles of System Dynamics. The second step or phase of the project is to complete the gaming interface of the developed model. For the design of the computer interface, several well-established principles in constructing informative and user-friendly gaming interfaces are utilized (2,6,7). The graphic-based spreadsheet software WingZ is used in developing the system (8,9). The third and final step is testing and validating the system. The validity of both the internal structure and model output is tested (10). Through the process of letting users play the game and provide feedback on the various aspects of the system, it is exposed to criticism, revised, exposed again in an iterative process until it proves to be valid. Both the system dynamics model behind the game

and the game itself are subjected to validity testing. Model is tested by running under various typical and extreme conditions and found to exhibit realistic behavior. The game is tested by playing with a wide range of player inputs, some of which are unlikely and extreme and by independent players' feedback as to the realism of the game. Just as the model is improved as a result of successive exposures to many players, a better understanding of the problem is achieved.

After analyzing the outputs of eight games (played by 8 players) and comparing them to the model's behavior, we can state that the interactive game shows a general behavior which, as expected, is similar to the model's behavior. In spite of some specific differences due to interactive decision making, the overall behavior of the game satisfies the idea or goal of constructing a game which is able to reflect the behavior of the players and real life software management projects.

II. GAME OVERVIEW

This is an educational game based on a system dynamics model of software project management, the fundamental structure of which is based on Abdel-Hamid & Madnick's model (1). The purpose of the game is to give the users the opportunity of having an interactive environment in which they can improve their understanding of the software development process and can learn how such a dynamic environment can be managed. It consists of three parts: "the simulation model", "the information system", and "the simulator controls". The model represents the structure of the software project management, including the human resources, software development, quality assurance and rework, testing, control and planning. The model generates dynamics of each of these components

over time as the player makes decisions. The information system reports the current state of the system and allows the player to review the history of the project. For example, one is be able to monitor the man-day expenditure of his/her project period by period, and receive reports on human resources, software development, etc. The controls allow one to make strategic and operational decisions to achieve his goals.

The Simulation Model:

The heart of the simulator is a simulation model of software project management and its environment. As indicated, the model has been extensively tested and calibrated. However, like any model, it is a simplification of reality. A number of factors have, of necessity, been omitted or simplified, just as a software project manager uses data about the development of project and cannot portray every circumstances or detail. Figure III provides an overview of the model. One should note that the sectors are highly interconnected. Decisions made in one sector may create opportunities and problems in other areas.

Information System:

The simulator contains an information system which allows player to monitor developments in all areas of the project. One has access to a number of variables which show the current status of his/her project. He or she may also review the history of his/her project in graphical form. As in many real situations, he/she is flooded with information and has to decide how to select the most useful data to assist him/her in making his/her decisions.

Simulator Controls:

Each decision period the player has the opportunity to make three decisions. These are:

1. % of man power to allocate for quality assurance.
2. % of man power to allocate for rework (the remaining man power will be used in development & testing).
3. staff additions/deletions

The purpose of the simulator is to give the player insight into the issues raised by a particular project; to illustrate the difficulties of coordinating operations and strategy in a software project environment; and to clarify the dynamic interconnections among a project's several sectors. More fundamentally, the simulation game is a laboratory in which one can systematically explore the consequences of various strategies without risking the possibility of bankruptcy and budget overrun.

The first time or two the player may want to try to succeed using some "guessed" strategies. In later trials he or she might wish to systematically vary aspects of his strategy to identify high-leverage policies. Most of his/her learning comes from understanding what went wrong with various strategies.

III. THE SIMULATION MODEL

As indicated earlier, the primary purpose of my model is to help us understand the process by which software systems are developed and managed. Notice that the focus is confined to the development phases of software production, extending only until the last phase of software development, namely, the testing phase. Not included in the model are the subsequent maintenance activities. My focus in this study is on the software development organization, i.e., project managers and software development professionals, and how their policies, decisions and actions affect the success or failure of software development. The definition of user requirements is therefore excluded from the model's boundary for the additional reason that it lies beyond the control of the

software development group. In addition, it is assumed that once requirements are fully specified and the architectural design phase is initiated, there will be no significant changes in users' requirements.

The model consists of seven major subsystems: human resource management, manpower allocation, software development, quality assurance and rework, system testing, controlling and planning. Figure III provides an overview of the model (For more information on the model, see (1)).

In my version of the model, I made certain modifications and simplifications. These were made to increase the speed of the model for use in the game. I removed certain variables and connections from the base model (1) that I thought did not have significant effect on the behavior of the model and I added variables that I thought were necessary in order to compensate for some simplifications. As a result of these changes, about 260 variables and 30 tables in the original model were reduced to 200 variables and 20 tables in my version. (Compare Appendix 1 and Tarek Abdel-Hamid & Stuart Madnick's model (1) for details)

Subsystems of the simulation model are graphically represented in terms of "level", "rate" and "convertor" variables. A "level" or"stock" is an accumulation, or an integration, over time of flows or changes that flow into and out of the level. Levels are represented by using rectangles. The flow variables are also called "rates". Rates are represented as valves (flowing into and out of levels). Flows will always originate somewhere and terminate somewhere. Sometimes the origin of flow is treated as essentially limitless, or at least outside the model's boundaries. In such a case the flow's origin is called a source. Similarly, when the destination of a flow is not of interest, it is called a sink. Both

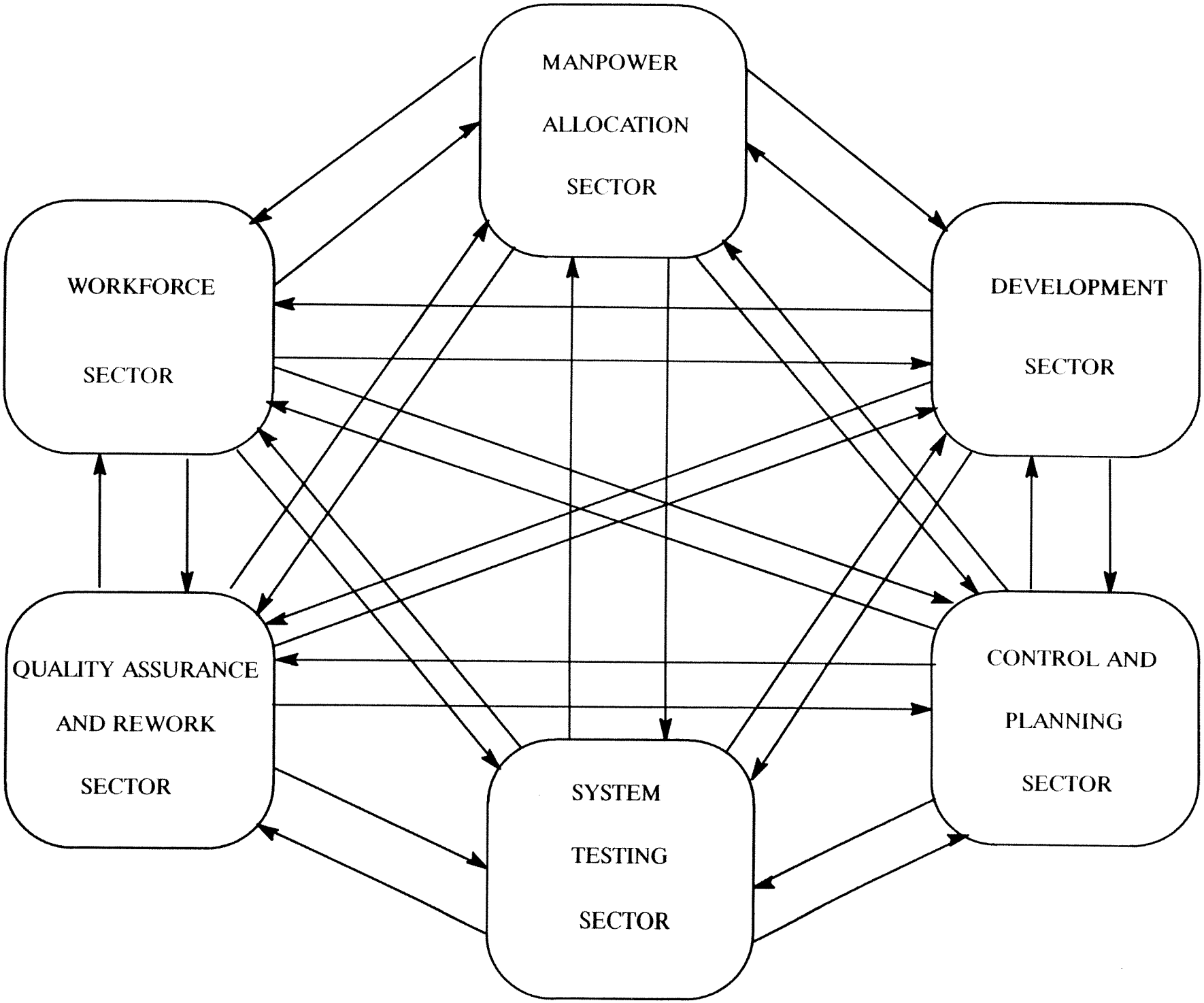

Figure III. Overview of the Simulation model

sources and sinks are shown as little "clouds". A third, intermediate type of variables are convertors. They are represented by circles. These variables represent how various information inputs are combined to yield certain statistics, decisions & actions (11). For more information about the model, see (1).

III. 1. Human Resource Management

This subsystem comprises the hiring, training, assimilation, and transfer of the project's human resources. Such actions are not carried out in a vacuum; they both affect and are affected by the other subsystems. Basic functions performed in this subsystem are: training of newly hired work force, assimilation of newly hired work force and determining work force level. The variables involved in my version of this sector are as follows:

WFNEW: New Workforce (People)
HIRERT: Hiring Rate (People/Day)
HIREDY: Hiring Delay (Days)
WFGAP: Workforce Gap (People)
NEWTRR: New Employees Transfer Rate Out (People/Day)
TRNFRT: Transfer Rate Of People Out Of Project (People/Day)
TRNSDY: Time Delay To Transfer People Out (Days)
ASIMRT: Assimilation Rate Of New Employees (People/Day)
ASIMDY: Average Assimilation Delay (Days)
DMPTRN: Daily Manpower For Training (Man-Days/Day)
TRPNHR: Number Of Trainers Per New Employee (Dimensionless)
WFEXP: Experienced Workforce (People)
EXPTRR: Experienced Employees Transfer Rate (People/Day)
QUITRT: Experienced Employees Quit Rate (People/Day)
AVEMPT: Average Employment Time (Days)
FTEXWF: Full-Time-Equivalent Experienced Workforce (Men)
CELNWH: Ceiling On New Hirees (Men)
MNHPXS: Most New Hirees Per Experienced Staff (Men/Men)
CELTWF: Ceiling On Total Workforce (People)
WFS: Workforce Sought (People)

TOTWF: Total Workforce Level (People)
FTEQWF: Full Time Equivalent Workforce (Equivalent People)
FRWFEX: Fraction of Workforce That Is Experienced (Dimensionless)

The variable for CMTRMD (Cumulative Training Man-Days) in the original model was removed in my model. For details, compare Abdel-Hamid & Madnick's model (1) and equations of my model given in Appendix 1.

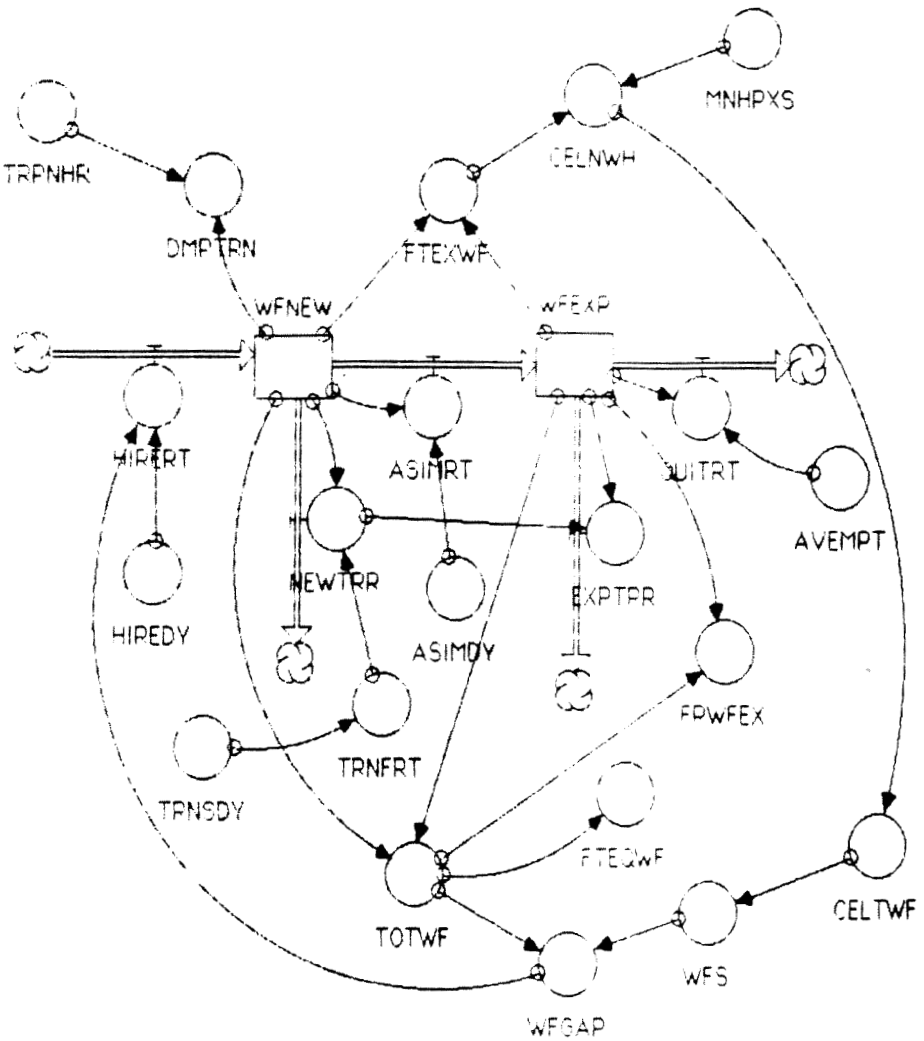

Figure III.1. Human Resources Management

III. 2. Manpower Allocation

This subsystem involves the allocation of workforce for different sectors of software project development. Main functions of this sector are: manpower allocation for quality assurance, impact of schedule pressure on manpower allocation for quality assurance, manpower allocation for rework. The variables involved in my version of this sector are as follows:

TOTDMP: Total Daily Manpower (Man-Days/Day)
ADMPPS: Average Daily Manpower Per Staff (Day/Day)
CUMMD: Cumulative Man-Days Expended (Man-Days)
DMPATR: Daily Manpower Available After Training (Man-Days/Day)
AFMPQA: Actual Fraction Of Manpower for Quality Assurance (Dimensionless)
PFMPQA: Planned Fraction Of Manpower for Quality Assurance (Dimensionless)
ADJQA: % Adjustment In PFMPQA (%)
DMPQA: Daily Manpower Allocated For Quality Assurance (Man-Days/Day)
DMPSWP: Daily Manpower For Software Production (Man-Days/Day)
DESECR: Desired Error Correction Rate (Errors/Day)
DESRWD: Desired Rework Delay (Days)
DMPRW: Daily Manpower Allocated For Rework (Man-Days/Day)
PRWMPE: Perceived Rework Manpower Needed Per Error (Man-Days/Error)
TARMPE: Time To Adjust PRWMPE (Days)
DMPDVT: Daily Manpower For Development/Testing (Man-Days/Day)
CHANGE1: Dummy Variable Used For Smoothing (Man-Days/Error)

Equation of the variable PFMPQA (Planned Fraction of Manpower for Quality Assurance) in the original model was modified in my version. The variables CMRWMD (Cumulative Rework Man-days), CMQAMD (Cumulative Quality Assurance Man-Days), CMDVMD (Cumulative Development Man-Days), QO (Quality Objective), TPFMQA (Table for PFMPQA), TADJQA (Table for % Adjustment in PFMPQA) in the original model were removed in my model.

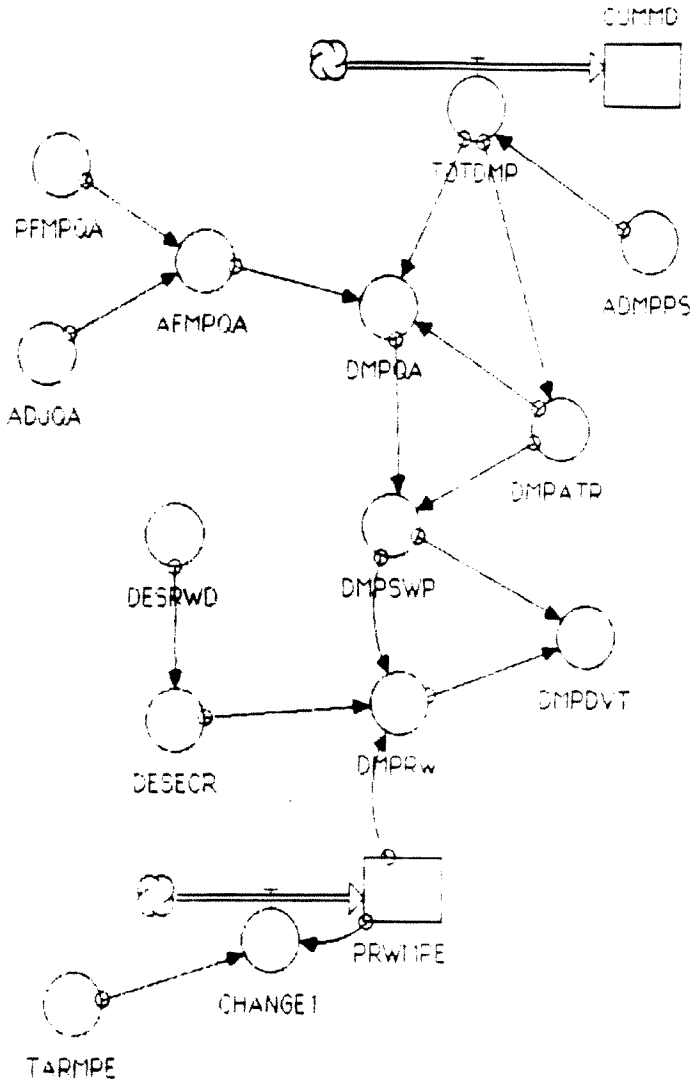

Figure III.2. Manpower Allocation

III. 3. Software Development

The software development process consists of the design and coding of the software product. Software project is defined as a number of "Tasks." Thus, the software development rate is a function of "Tasks per day," software developed of "Tasks" developed, and software development productivity of "Tasks per man-day." After manpower allocations are made for training, quality assurance, and rework activities, the remaining bulk of the available manpower resource is allocated to the development of software product. This allocation continues until it is perceived that most of the software development tasks are completed, at which point the system testing phase begins and manpower is allocated for testing. The variables involved in my version of this sector are as follows:

SDVRT1,SDVRT2: Software Development Rate (Tasks/Day)
DMPSDV: Daily Manpower For Software Development (Man-Days/Day)
FREFTS: Fraction Of Effort For System Testing (Dimensionless)
SDVPRD: Software Development Productivity (Tasks/Man-Day)
POTPRD: Potential Productivity (Tasks/Man-Day)
ANPPRD: Average Nominal Potential Productivity (Tasks/Man-Day)
NPWPEX: Nominal Potential Productivity Of Experienced Employee (Tasks/Man-Day)
NPWPNE: Nominal Potential Productivity Of New Employee (Tasks/Man-Day)
MPPTPD: Multiplier To Potential Productivity Due To Learning
MPDMCL: Multiplier To Productivity Due To Motivation And Communication Losses (Dimensionless)
COMMOH: Communication Overhead (Dimensionless)
NFMDPJ: Nominal Fraction Of A Man-Day On Project (Dimensionless)
AFMDPJ: Actual Fraction Of A Man-Day On Project (Dimensionless)
WRADJR: Work Rate Adjustment Rate (1/Day)
WKRADY: Work Rate Adjustment Delay (Days)
NWRADY: Normal Work Rate Adjustment Delay (Days)
EWKRTS: Effect Of Work Rate Sought (Dimensionless)
WKRTS: Work Rate Sought (Dimensionless)

MAXMHR: Maximum Boost In Man-Hours (Dimensionless)
PBWKRS: % Boost In Work Rate Sought (%)
MDHDL: Man-Days That Will Be Handled Or Absorbed (Man-Days)
CTRLSW: Control Switch...Allows Us To Test Policy Of No Overwork
EXSABS: Man-Days Excesses That Will Be Absorbed (Man-Days)
TEXABS: Table For EXSABS (Dimensionless)
MAXSHR: Maximum Shortage In Man-Days That Can Be Handled (Man-Days)
WTOVWK: Willingness To Overwork (0 or 1)
BRKDTM: Time Of Last Exhaustion Breakdown
BREAKDOWN: Accumulation Rate Of Exhaustion For Breakdown
SW: Switch Used To Control Breakdown Rate
RLXTMC: Variable That Controls Time To De-Exhaust
DEEXHAUST: First Control Rate For De-Exhaustion (Dimensionless)
DISCHARGE: Second Control Rate For De-Exhaustion (Dimensionless)
OVWDTH: Overwork Duration Threshold (Days)
NOVWDT: Nominal Overwork Duration Threshold (Days)
MODTEX: Effect Of Exhaustion On Overwork Duration Threshold
EXHLEV: Exhaustion Level (Exhaust Units)
RIEXHL: Rate Of Increase In Exhaustion Level (Exhaust Units/Day)
RDEXHL: Rate Of Depletion In Exhaustion Level (Exhaust Units/Day)
EXHDDY: Exhaustion Depletion Delay Time (Days)
MXEXHT: Maximum Tolerable Exhaustion (Exhaust Units)

The equations of the following variables in the original model were simplified in my version: SDVRT (Software Development Rate), EWKRTS (Effect of Work Rate Sought), PBWKRS (% Boost in Work Rate Sought), MDHDL (Man-Days That Will be Handled or Aborbed), WTOVWK (Willingness to Overwork), BRKDTM (Time of Last Exhaustion Breakdown), RLXTMC (Variable that Controls Time to De-Exhaust). The following variables were removed from the original model: TFEFTS (Table for Fraction of Effort for System Testing), TMPTPD (Table for multiplier to Potential Productivity due to Motivation and Communication Losses), TCOMOH (Table for Communication Overhead), TNWRAD (Table for Normal Work Work Rate Adjustment Delay), TMODEX (Table for Exhaustion on Overwork Duration Threshold), TNOWDT (Table for Nominal Overwork Duration Threshold).

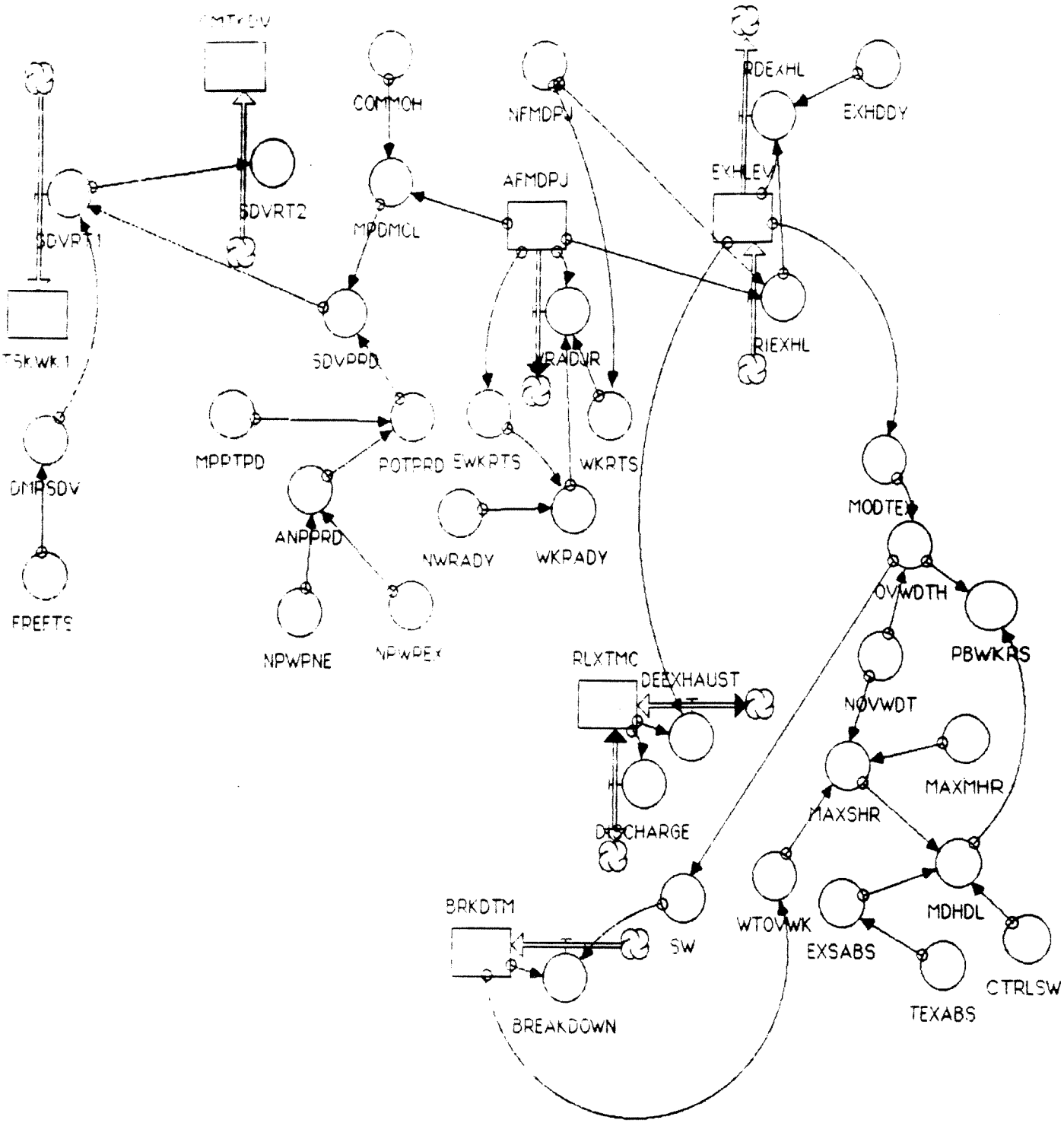

Figure III.3. Software Development

III. 4. Quality Assurance and Rework

The development of software system involves a series of production activities where the opportunities for interjection of human fallibilities are enormous. Errors may begin to occur at the inception of the process where the objectives of the software system may be erroneously or imperfectly specified, as well as during the later design and development stages where these objectives are mechanized. The basic quality for software is that it performs its functions in the manner that was intended by its architects. To achieve this quality, the final product must contain a minimum of mistakes in implementing their intentions as well as being void of misconception about the intentions themselves. Because of human inability to perform with perception, software development is accompanied by a quality assurance. In our model, this subsystem involves the generation, detection and correction of errors during the development phase. The variables involved in my version of this sector are as follows:

QART1,QART2: For Quality Assurance Rate (Tasks/Day)
TSKWK1,TSKWK2: Tasks Worked (Tasks)
TSKWK: Total Tasks Worked (Tasks)
AQADLY: Average Delay For Quality Assurance (Days)
CUMTQA: Cumulative Tasks Quality Assured (Tasks)
ANERPT: Average # Of Errors Per Task (Errors/Task)
ERRDSY: Error Density (Errors/KDSI)
ERRDRT: Error Detection Rate (Errors/Day)
ERRSRT: Error Escape Rate (Errors/Day)
PTDTER: Potentially Detectable Errors (Errors)
ERRGRT: Error Generation Rate (Errors/Day)
ERRPTK: Errors Per Task (Errors/Task)
NERPTK: Nominal # Of Errors Committed Per Task (Errors/Task)
NERPK: Nominal # Of Errors Committed Per KDSI (Errors/KDSI)
MERGSP: Multiplier To Error Generation Due To Schedule Pressure
MERGWM: Multiplier To Error Generation Due To Workforce Mix
DTCERR: Detected Errors (Errors)
RWRATE: Rework Rate (Errors/Day)

RWMPPE: Rework Manpower Needed Per Error (Man-Days/Error)
NRWMPE: Nominal Rework Manpower Needed Per Error (Man-Days/Error)
DSIPTK: DSI Per Task

The equations of the following variables in the original model were simplified in my version: QART (Quality Assurance Rate), TSKWK (Tasks Worked),ERRSRT (Error Escape Rate). The following variables were removed from the original model: QAMPNE (Quality Assurance Manpower Needed to Detect Average Error), NQAMPE (Nominal Quality Assurance Manpower Needed to Detect Average Error), MDEFED (Multiplier to Detection Effort due to Error Density), TMDFED (Table for MDEFED), PERDRT (Potential Error Detection Rate), CMERD (Cumulative Errors Detected), PRCTDT (Percent Errors Detected), CMERES (Cumulative Errors That Escaped), TNERPK (Table for Nominal # of Errors Committed Per KDSI), TMEGSP(Table for Multiplier to Error Generation due to Schedule Pressure), TMEGWM (Table for Multiplier to Error Generation due to Workforce Mix), CUMERG (Cumulative Errors Generated Directly During Working), TNRWME (Table for Nominal Rework Manpower Needed Per Error), CMRWED (Cumulative Reworked Errors During Development).

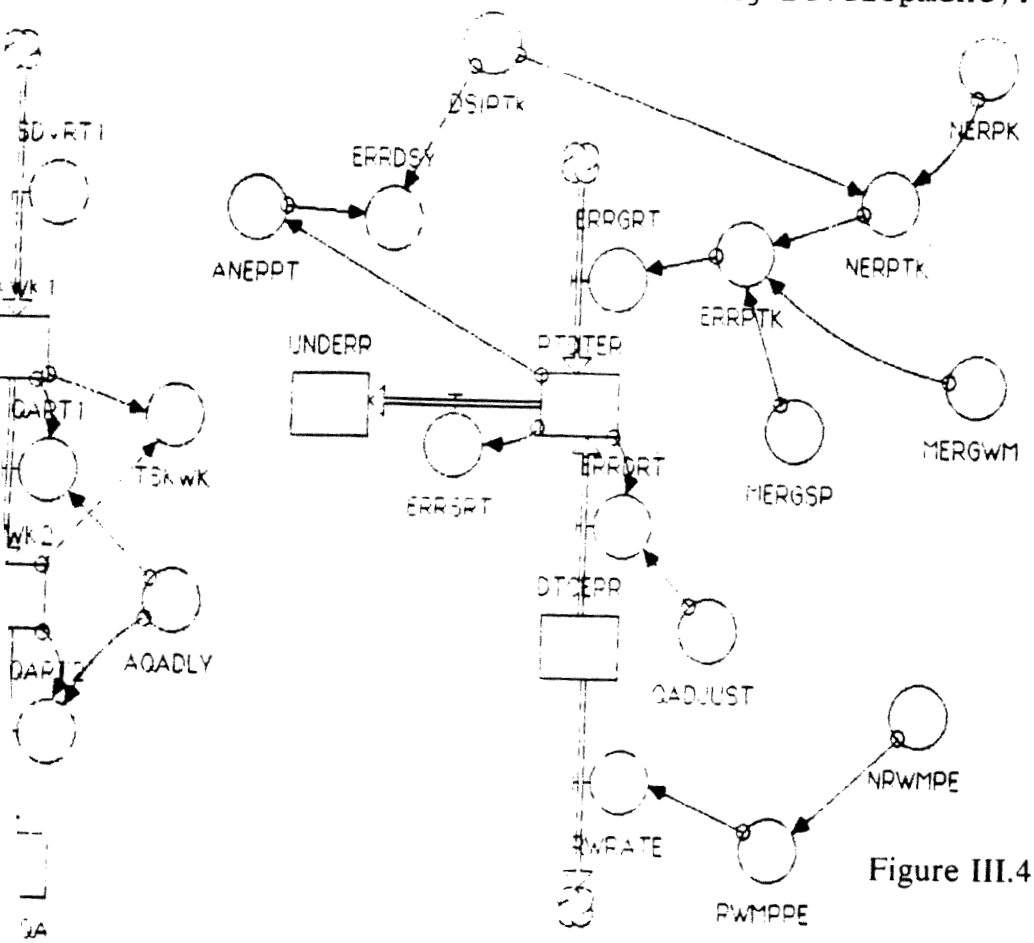

Figure III.4. Quality Assurance and Rework

III. 5. System Testing

Errors that quality assurance fails to detect while the software is being designed and coded and bad fixes from faulty rework remain undetected until the system testing phase. I will assume that all such errors will be detected and corrected at the system testing phase. This sector models two sets of processes: the growth of the undetected error populations and the system testing that results in the detection and correction of those errors. The variables involved in my version of this sector are as follows:

UNDERR: Undetected Errors (Errors)
RGNRT: Error Regeneration Rate (Errors/Day)
CORRECT: Correction Rate (Errors/Day)
TSAEDS: Time To Smooth Error Density (Days)
UNDERRDSY: Undetected Error Density (Errors/Task)
SMTERRDSY: Smoothed Undetected Error Density (Errors/Task)
DMPTST: Daily Manpower For Testing (Man-Days/Day)
CMTSMD: Cumulative Testing Man-Days (Man-Days)
TSRATE: Testing Rate (Tasks/Day)
TMPNPT: Testing Manpower Needed Per Task (Man-Days/Task)
TSTOVH: Testing Effort Overhead (Man-Days/KDSI)
TMPNPE: Testing Manpower Needed Per Error (Man-Days/Error)
PTKTST: % Of Tasks Tested (%)
CUMTKT: Cumulative Tasks Tested (Tasks)

The following variables were removed from the original model: UDAVER (Undetected Active Errors), AEGRT (Active Errors Generation Rate), BDFXGR (Bad Fixes Generate Rate), PBADFX (Percent Bad Fixes), FRAERR (Fraction of Escaping Errors That Will Be Active), TFRAER (Table for FRAERR), AERGRT (Active Errors Regeneration Rate), MAERED (Multiplier to Active Error Regeneration due to Error Density), TMERED (Table for MAERED), AERRDS (Active Error Density), AERRRT (Active Errors Retiring Rate), AERRFR (Active Errors

Retiring Fraction), TERMFR (Table for AERRFR), DCRTAE (Detection/Correction Rate of Passive Errors), UDPVER (Undetected Passive Errors), PEGRT (Passive Errors Generation Rate), DCRTPE (Detect/Correct Rate of Passive Errors), CMRWET (Cumulative Errors Reworked in Testing Phase), ALESER (All Errors That Escaped and Were Generated), PERRDS (Passive Error Density), ALLERR (All Errors), ALLRWK (All Errors Reworked in Development and Testing). The variables UNDERR,RGNRT,CORRECT,UNDERRDSY,SMTERRDSY were added to the original model.

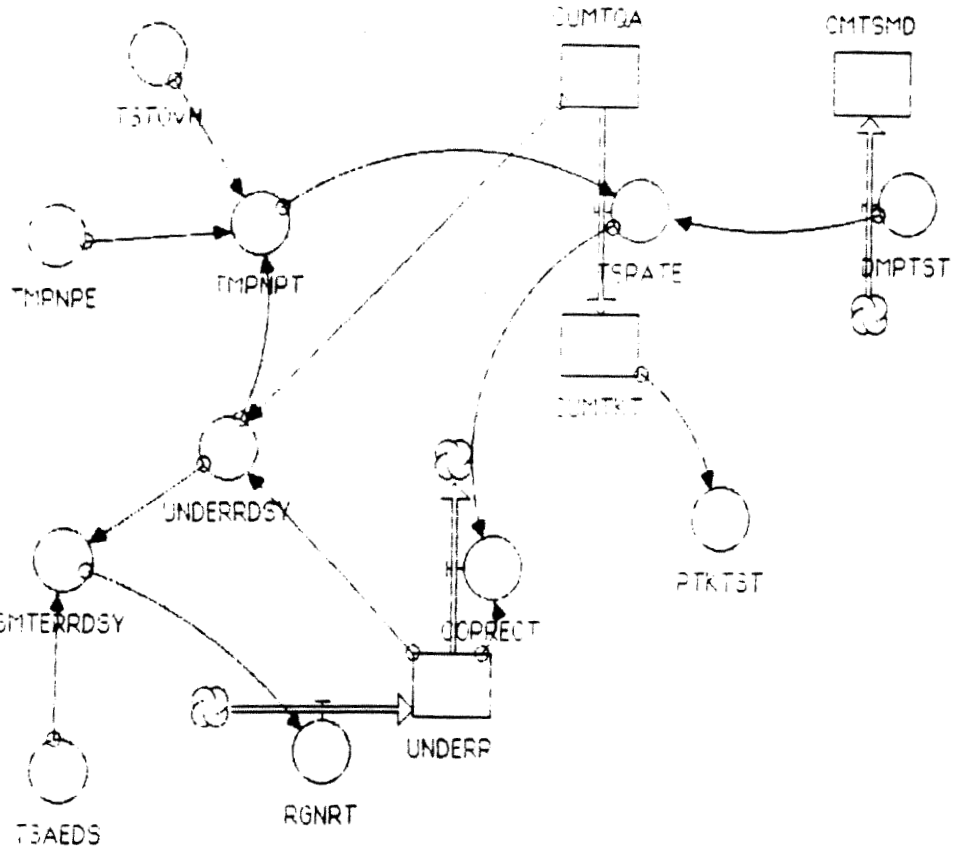

Figure III.5. System Testing

III. 6. Controlling

A comparison of where the project is versus where it should be (according to plan) is a control activity captured within this subsystem. Three elements that are included in the control function of software project management are measurement (detection of what is happening in the activity being controlled), evaluation (assessment of its significance, by comparing information on what is actually happening with some standard or expectation of what should be happening and communication) report of what has been measured and assessed, so that behavior can be altered if the need for doing so is indicated. The variables involved in my version of this sector are as follows:

CMTKDV: Cumulative Tasks Developed (Tasks)
PJBAWK: % Of Job Actually Worked (%)
PJDPRD: Projected Development Productivity (Tasks/Man-Day)
MDPRNT: Man Days Perceived Needed For New Tasks (Man-Days)
MDPNRW: Man Days Perceived Needed For Reworking Already Detected Errors (Man-Days)
ASSPRD: Assumed Productivity (Tasks/Man-Day)
PRDPRD: Perceived Development Productivity (Tasks/Man-Day)
WTPJDP: Weight To Projected Development Productivity(Dimensionless)
MPWDEV: Multiplier To Productivity Weight Due To Development
MPWREX: Multiplier To Productivity Weight Due To Resource Expenditure (Dimensionless)
MDPNNT: Man Days Perceived Still Needed For New Tasks (Man-Days)
TMDPSN: Total Man Days Perceived Still Needed (Man-Days)
MDPNTS: Man Days Perceived Still Needed For Testing (Man-Days)
TSTPRM: Tasks Remaining To Be Tested (Tasks)
PRTPRD: Perceived Testing Productivity (Tasks/Man-Day)
TSTSPD: Time To Smooth Testing Productivity (Days)
PLTSPD: Planned Testing Productivity (Tasks/Man-Day)
ACTSPD: Actual Testing Productivity (Tasks/Man-Day)
SELECT: Variable To Make A Selection Between PLTSPD And ACTSPD (Tasks/Man-Day)

DUMMYRATE: Dummy Rate To Adjust PRTPRD (Tasks/Man-Day*Day)
PMDSHR: Perceived Shortage In Man-Days (Man-Days)
SHRRPT: Shortage Reported (Man-Days)
MDRPTN: Man Days Reported Still Needed (Man-Days)
UNDJTK: Undiscovered Job Tasks (Tasks)
RTDSTK: Rate Of Discovering Tasks (Tasks/Day)
PUTDPD: Percent Of Undiscovered Tasks Discovered Per Day (1/Day)
RJBSZ: Real Job Size In Tasks (Tasks)
PJBPWK: % Of Job Perceived Worked (%)
RTINCT1,RTINCT2: Rate Of Incorporating Discovered Tasks Into Project (Tasks/Day)
DISC1,DISC2: Tasks Discovered (Tasks)
TKDSCV: Total Tasks Discovered (Tasks)
DLINCT: Average Delay In Incorporating Discovered Tasks (Days)
PJBSZ: Currently Perceived Job Size (Tasks)
TSKPRM: New Tasks Perceived Remaining (Tasks)
PSZDCT: Perceived Size Of Discovered Tasks In Man Days (Man-Days)
RSZDCT: Relative Size Of Discovered Tasks (Dimensionless)
FADHWO: Fraction Of Additional Tasks Adding To Man-Days
MSZTWO: Maximum Relative Size Of Additions Tolerated Without Adding To Project's Man-Days
IRDVDT: Rate Of Increase In Development Man-Days Due To Discovered Tasks (Man-Days/Day)
TSZZMD: Planned Testing Size In Man-Days...Before We Start Testing (Man-Days)
IRTSDT1,IRTSDT2: Rate Of Increase In Testing Man Days Due To Discovered Tasks (Man-Days/Day)
JBSZMD: Total Job Size In Man Days (Man-Days)
ARTJBM: Rate Of Adjusting The Job Size In Man-Days (Man-Days/Day)
DAJBMD: Delay In Adjusting Job's Size In Man Days (Days)
MDRM: Man Days Remaining (Man-Days)
SCHPR: Schedule Pressure (Dimensionless)

The following variables were removed from the original model: TMPDEV (Table for Multiplier to Productivity due Weight due to Development), TMPREX (Table for Multiplier to Productivity Weight

due to Resource Expenditure), PRTPTC (% of Tasks Reported Complete), RPTDLY (Reporting Delay), PDEVRC (% Development Perceived Complete), TPUTDD (Table for Percent of Undiscovered Tasks Discovered Per Day), TFAHWO (Table for Fraction of Additional Tasks Adding to Man-Days), TDAJMD (Table for Delay in Adjusting Job's Size in Man-Days). The equations of the following variables in the original model were simplified in my version: PRTPRD (Perceived Testing Productivity), RTINCT (Rate of Incorporating Discovered Tasks into Project), TSZZMD (Planned Testing Size in Man-Days...Before We Start Testing).

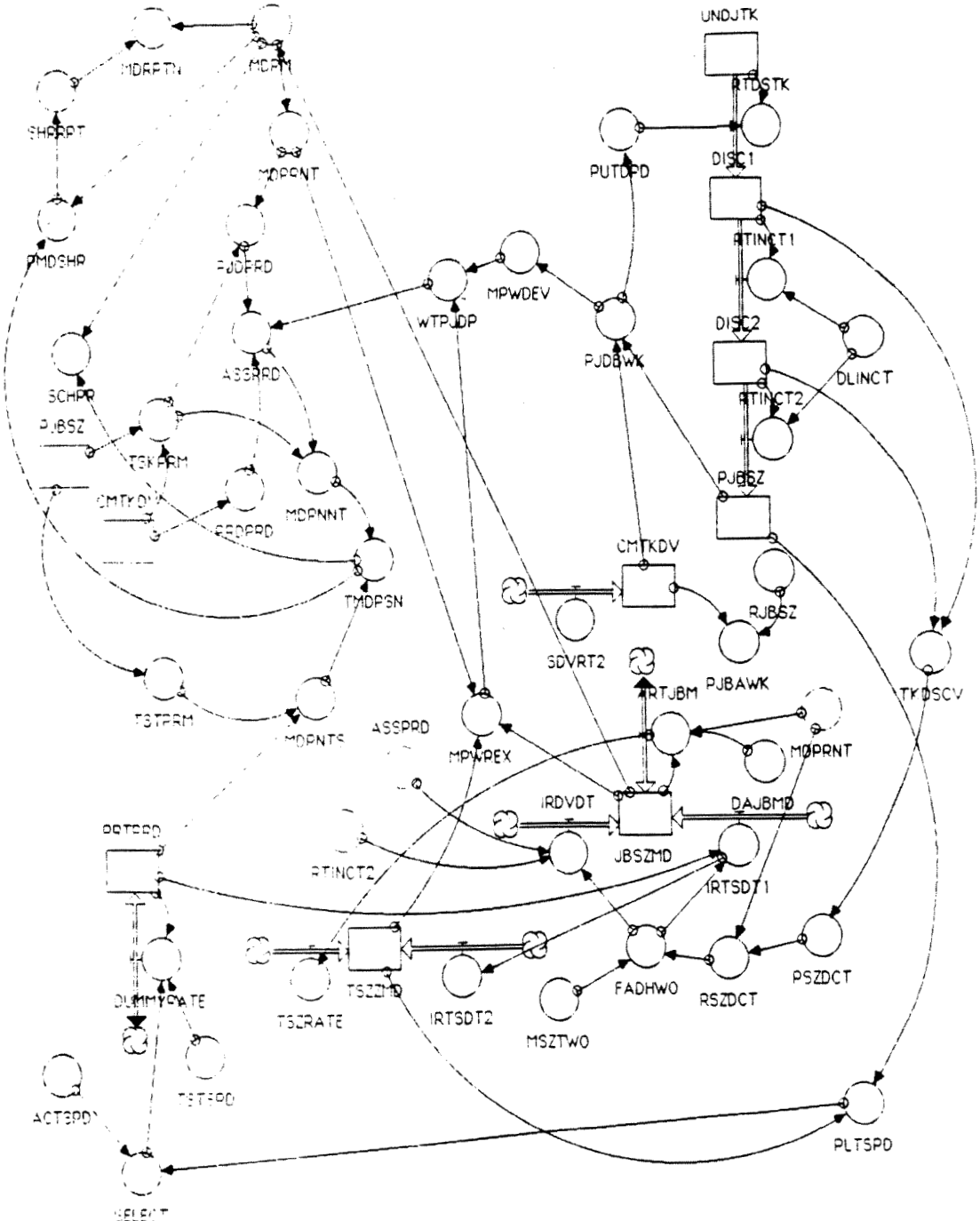

Figure III.6. Controlling

III. 7. Planning

In this subsystem, initial project estimates are made to start the project, and then those estimates are revised, when necessary, throughout the project's life. For example, to handle a project that is perceived to be behind schedule, a manager can hire more people, extend the schedule, or do a little of both. Main functions are work force level adjustments, schedule stability and completion date determination. The variables involved in my version of this sector are as follows:

TIMEPR: Time Perceived Still Required (Days)
INDCDT: Indicated Completion Date
SCHCDT: Scheduled Completion Date
CHANGE3: Rate To Make Adjustment Between INDCDT And SCHCDT (1/Day)
SCHADT: Schedule Adjustment Time (Days)
TIMERM: Time Remaining (Days)
WFINDC: Indicated Workforce (People)
WFNEED: Workforce Level Needed (People)
WCWF: Willingness To Change Workforce (Dimensionless)

The following variables were removed from the original model: TSHADT (Table for Scheduled Adjustment Time), WCWF1 (Willingness to Change Workforce (1)), WCWF2 (Willingness to Change Workforce (2)), TWCWF1 (Table for WCWF1), TWCWF2 (Table for WCWF2), MXTLDC (Maximum Tolerable Completion Date), MXSCDX (Maximum Schedule Completion Date Extension).

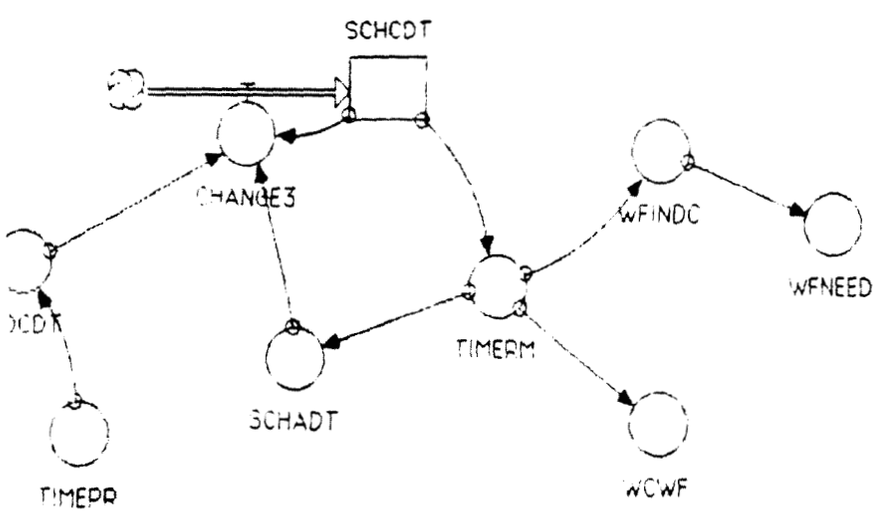

Figure III.7. Planning

III. 8. Initialization

The variables used in initialization are as follows:

RBDSI: Real Job Size In DSI

UNDEST: Tasks Underestimation Fraction

PJBDSI: Perceived Job Size In DSI

TOTMD: Total Man Days

UNDESM: Man-Days Underestimation Fraction

DEVMD: Development Man Days

DEVPRT: % Of Effort Assumed Needed For Development

TSTMD: Testing Man Days

WFSTRT: Team Size At Beginning Of Design (Men)

INUDST: Initial Understaffing Factor (Dimensionless)

TDEV: Total Development Time (Days)

TEAMSZ: Team Size (Man-Days/Day)

The following variables were removed from the original model: TOTMD1 (Total Man-Days), MDSWCH (Switch for TOTMD1...0 or 1), SCHCOM (Schedule Compression Factor), SCSWCH (Switch for TDEV1...0 or 1), TDEV1 (Time to Develop).

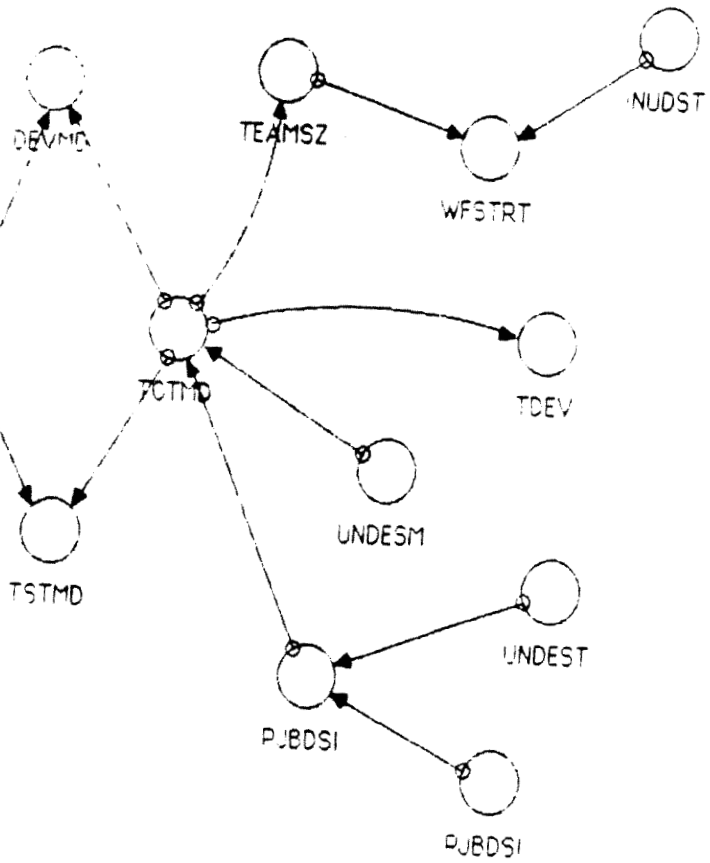

Figure III.8. Initialization

IV. BEHAVIOR OF THE SIMULATION MODEL

In this section, my model will be used to study the implications of an array of managerial actions, policies, and procedures pertaining to the development of software. Two models, one with underestimation in project size in terms of number of tasks, and one without underestimation will be simulated.

i) With Underestimation

The project is initially perceived to be less than its true size. As the project develops, "Undiscovered Job Tasks" are progressively discovered as our knowledge of what software is intended to do increases. Behavior of the model is shown in Fig. IV.1.a and IV.1.b. The rate at which "Perceived Job Size" rises remains low for a significant portion of the development phase, before it starts to accelerate rapidly (Fig. IV.1.a, Curve 4). As the additional tasks are discovered and project members start realizing that the project's scope is larger than what has been expected, adjustments are made in the project's plan to accommodate the additional work load (Fig. IV.1.a, Curves 1 & 2). As Figure IV.1.a indicates, both the "Job Size" and the "Scheduled Completion Date" are adjusted upwards. First, the adjustments prove to be inadequate to fully accommodate the additional work load. Therefore, a second adjustment is made to scheduled completion date.

If we look at the daily manpower and workforce distribution throughout the project in Figure IV.1.a, there is an upward trend in both new workforce and experienced workforce (Curves 4 & 5), which means that at the initial phases of the project, new workforce joins to the project while existing ones get experienced.

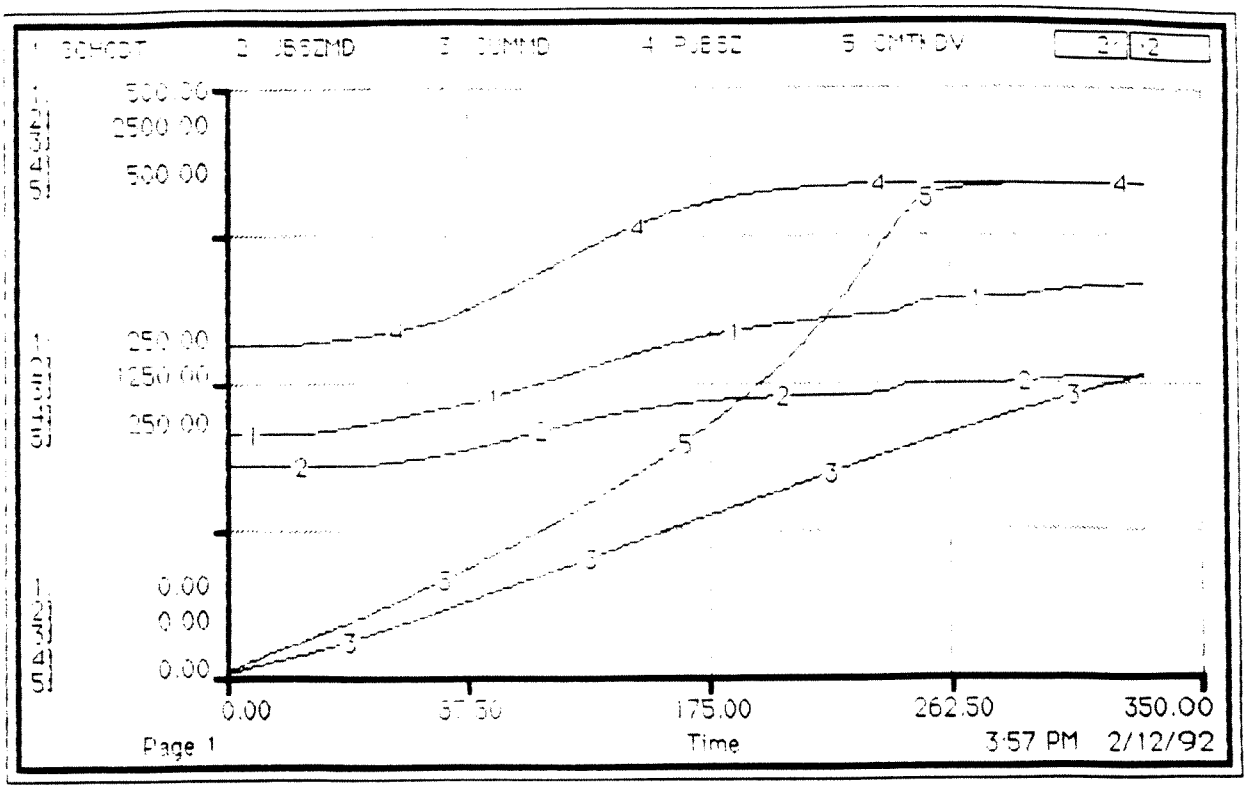

Figure IV.1.a

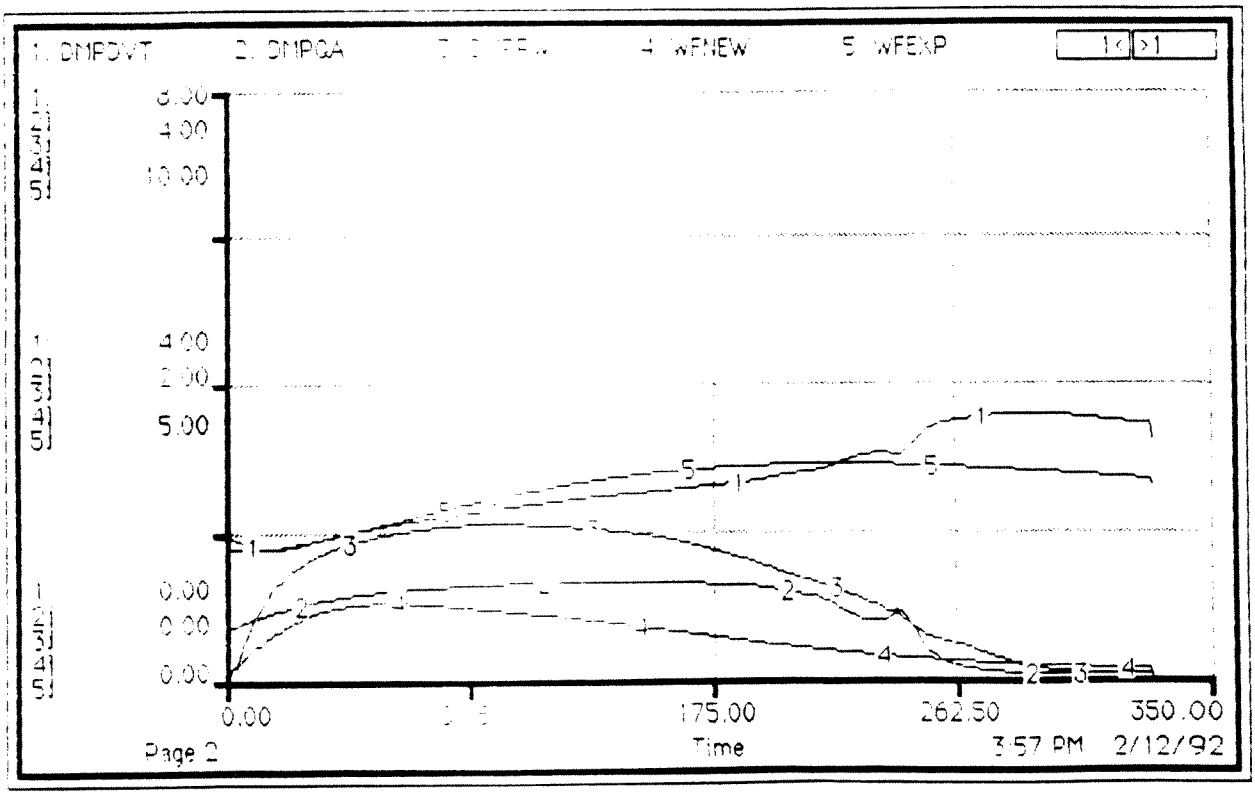

Figure IV.1.b

Figure IV.1. Model Output (With Underestimation)

Also, most of the manpower is allocated for task development, quality assurance and rework (Curves 1,2 & 3). When the time is around 75 days the model stops adding new workforce to the project. When time is about 100 days, daily manpower for rework (Curve 3), and when it is about 140 days, daily manpower for quality assurance (Curve 2) reach their peak. After completion of 90% of the development of the tasks, manpower is allocated for testing, and since quality assurance is a separate activity prior to testing, manpower allocated for quality assurance rapidly drops to zero. Since the development phase is already finished and the project is in testing phase, toward the end of the project all man power is allocated to rework and testing. When 99% of the system testing is done, the project is considered to be finished. At the end of the project, cumulative tasks developed is equal to project size in terms of number of tasks (Fig. IV.1.a, Curves 4 & 5) and cumulative man days expended is equal to project size in man days (Fig IV.1.a, Curves 2 & 3).

ii) Without Underestimation

In this case, since the project is initially perceived as its true size, there are no new tasks to be discovered and therefore no adjustment in job size (Fig. IV.2.a, Curve 4). The model generates almost the same behavior pattern as the one with underestimation does (Fig. IV.2.a). Manpower is allocated to development, quality assurance and rework at the initial phases and new workforce is added. Then, when time is about 200 days, 90% of the task development is finished. Therefore manpower allocation is shifted from quality assurance and development to testing. Again, after the first additions to the workforce at the beginning of the project, there is no other significant hiring (Fig. IV.2.b). As in the case with underestimation, at the end of the project, cumulative man

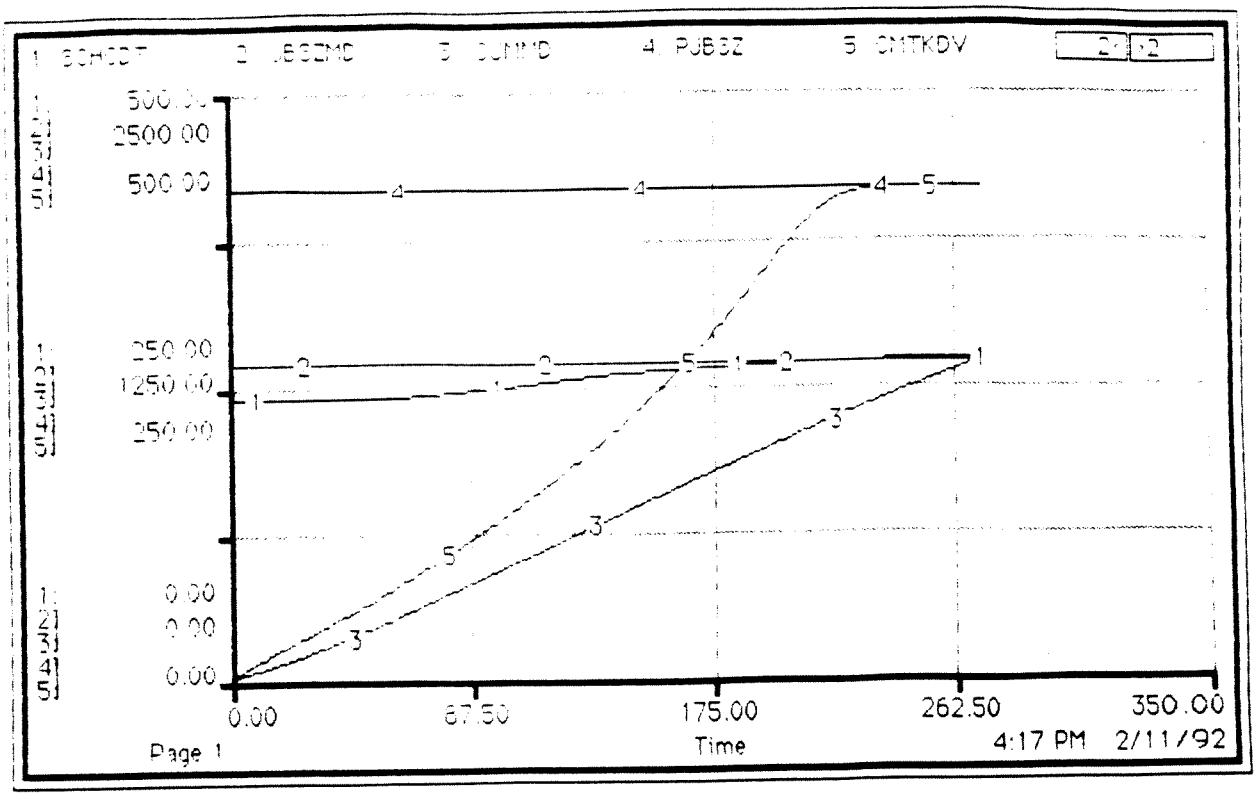

Figure IV.2.a

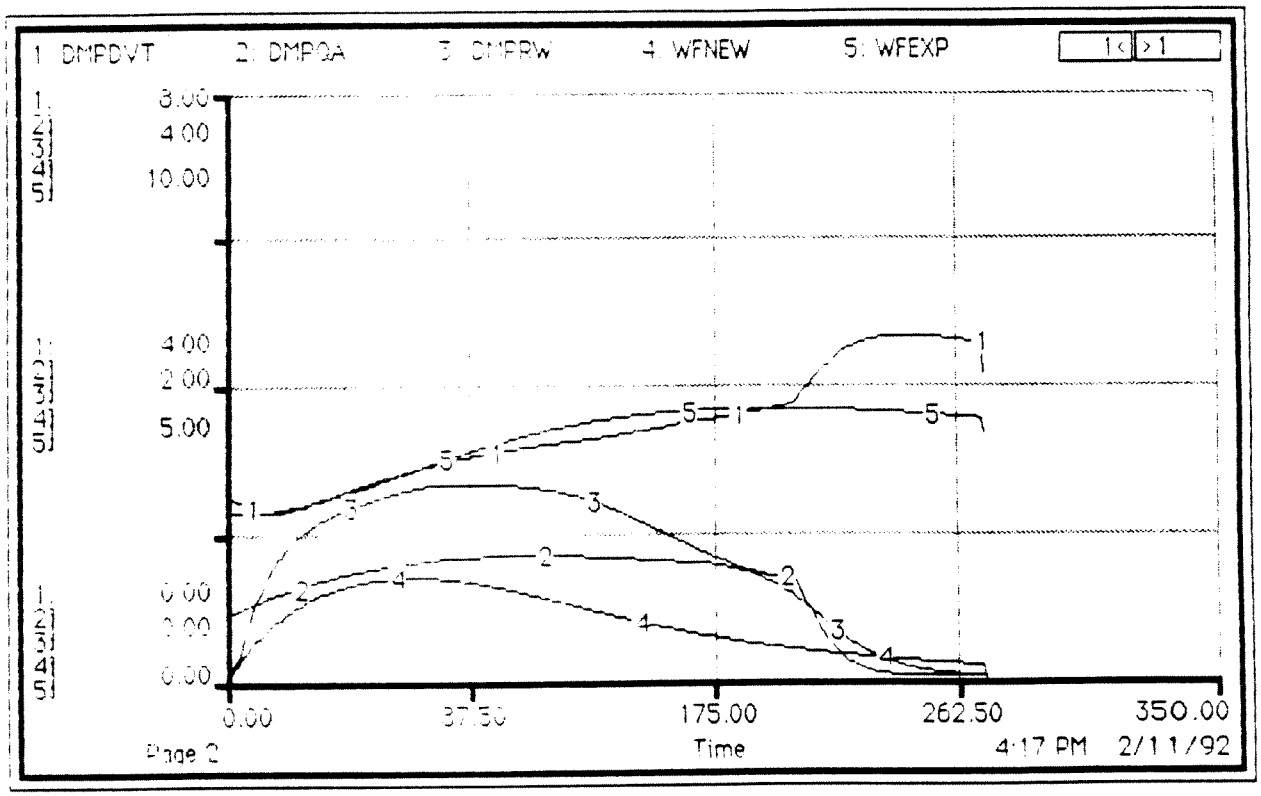

Figure IV.2.b

Figure IV.2. Model Output (Without Underestimation)

days expended is equal to job size in man days (Fig. IV.2.a, Curves 1 & 2) and cumulative tasks developed is equal to project size in terms of number of tasks (Fig. IV.1.a, Curves 4 & 5).

V. THE INTERACTIVE SIMULATION GAME

For the purpose of constructing the game, user inputs are incorporated into the model. By replacing or modifying some of the original equations in the model, we give the player the opportunity of making three decisions: % of manpower allocated for quality assurance, % of manpower allocated for rework (the remaining being allocated for development and testing) and staff additions/removals. This way, the player can see how the model is responding to certain decisions, how one change in one subsystem creates opportunities or problems in another one.

The interactive game is developed in five phases. The first phase is the selection of the software. For this purpose, a graphic-based IBM-PC spreadsheet software WingZ is used. I could not use Stella II in game development, because it does not have any graphical and interactive features needed to do the game interface. It could have been easier to develop the game by using a Macintosh software, but the computers used in the department were almost all IBM-PC's.

As the second phase, equations in Stella II are coded in WingZ script and the model is verified by running it under WingZ and obtaining the same behavior that is obtained in Stella II (About 25 variables are tested to compare the behavior of the model in Stella to the behavior in WingZ).

In the third phase, some equations are modified to convert the model to a user-interactive game.

In order to incorporate % of manpower allocated for quality assurance into the model as an interactive player decision, the

The main object of the game is to finish the project within certain limits of time(days) and budget(man-days). Remember this is not a race; player is not trying to finish game as early as possible. As long as he/she finishes it within the limits, he/she should consider his/her performance successful. For example, for the small size project, the time limit is 500 days and the budget limit is 2500 man-days.

Making Decisions

The game requires player to make three decisions. Click on "make decision" button (left button on the main screen in Fig. VI.1) when player is ready to input his/her decisions. Having done that, a scrolling window (Figure VI.2.a) appears in the middle of the screen. Player is asked to enter the percentage of daily manpower (in man-days) to be allocated for quality assurance. The number that is shown in the text is calculated by subtracting daily manpower allocated for training from the total daily manpower. This allocation for training is done by the model; the remaining figure shows player the total daily manpower that he/she can use for his/her other decisions. The number highlighted in the wheel is player's previous decision. If he/she simply wants to repeat his/her previous decision, just click "ok". Player can either scroll using the mouse or type in his/her new decision.

After clicking "ok" or using the "Return" keyboard button, a new window (Figure VI.2.b) appears in the same location. PLayer is then ready to enter his/her request for percentage of man-power to be allocated for rework. The value in the text is calculated by subtracting the amount that player allocated for quality assurance from the manpower value of the first decision window. After he/she makes this decision, the remaining manpower will be allocated to development and testing. Finally, the allocation between development and testing is done automatically by model. Player should also note that in the model, testing does not start until 90% of the development is completed.

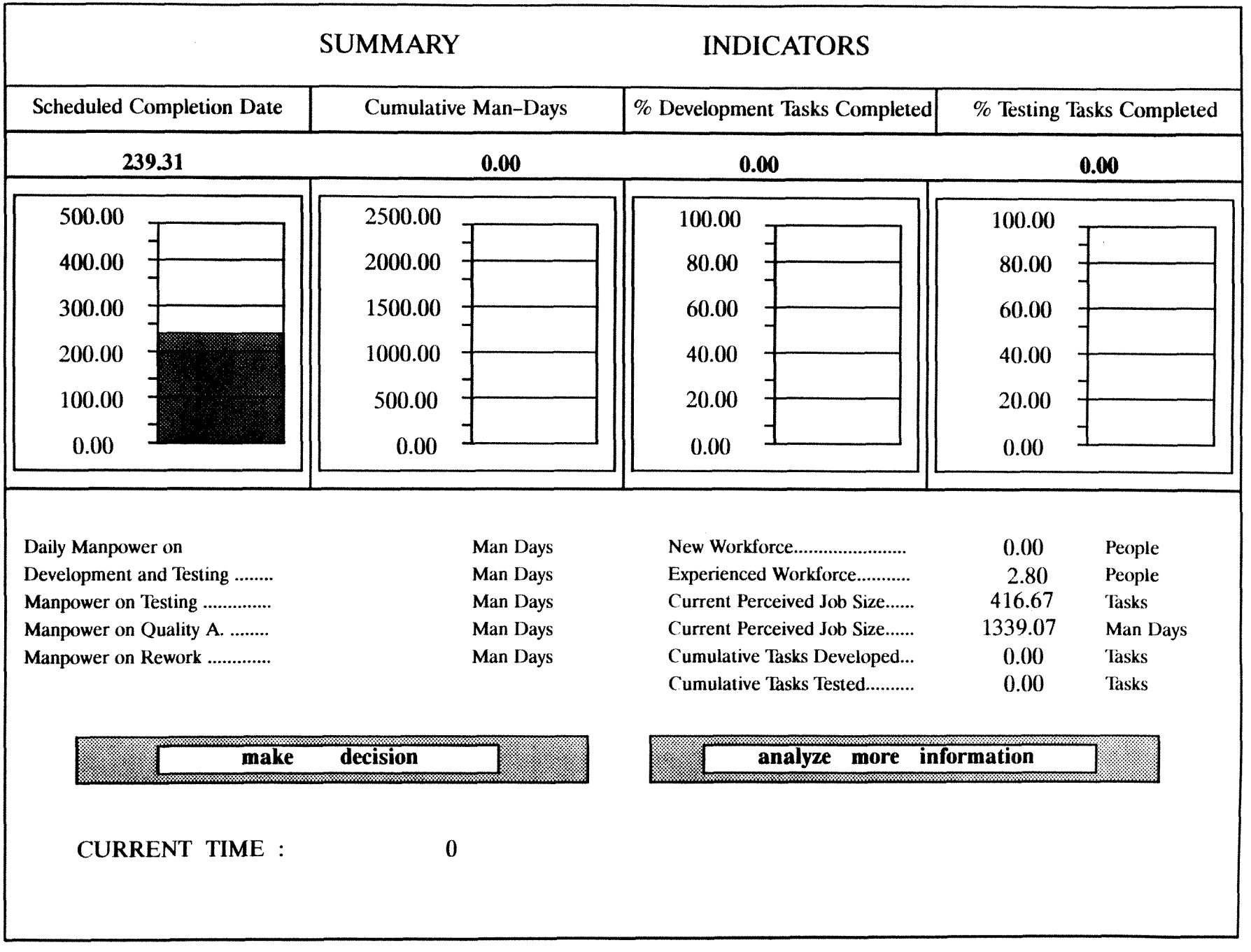

Figure VI.1. Main Screen

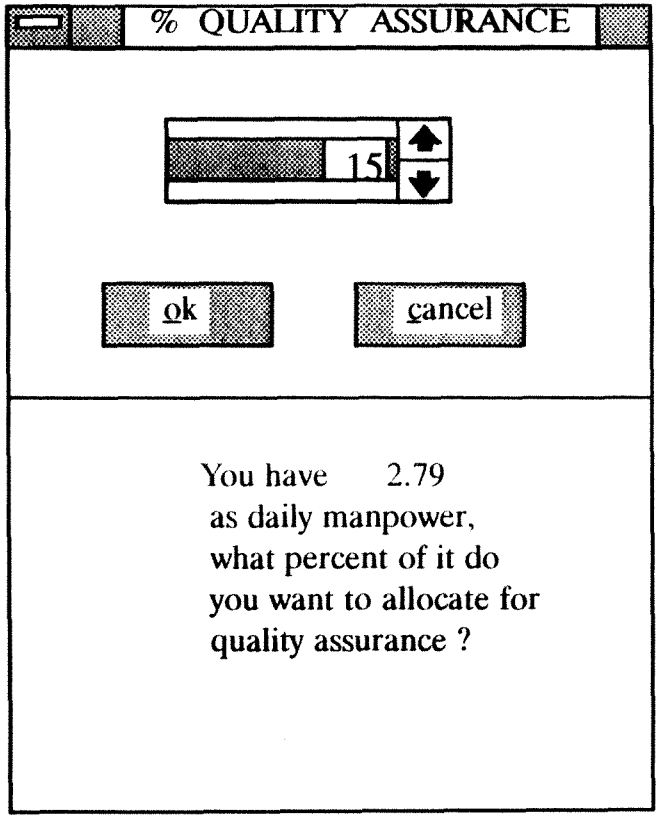

Figure VI.2.a

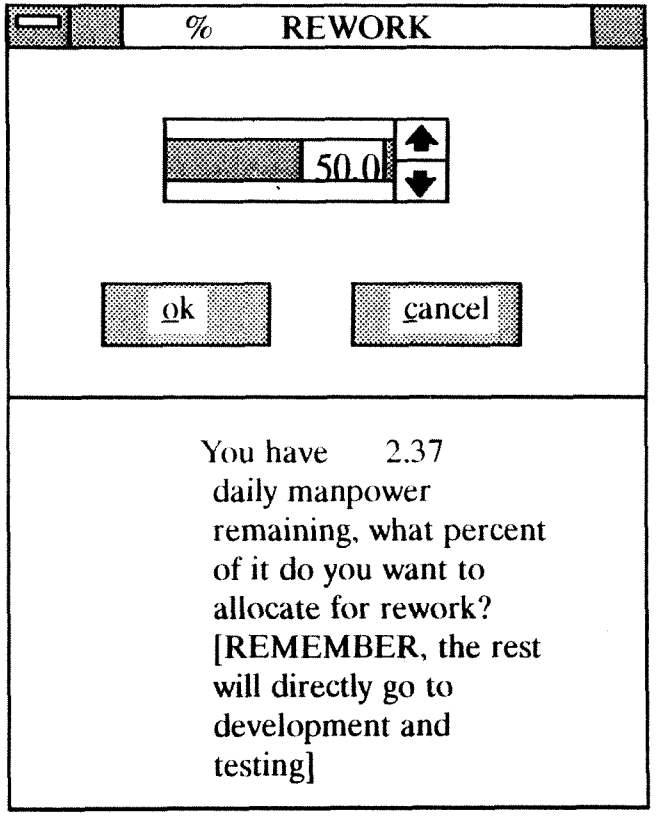

Figure VI.2.b

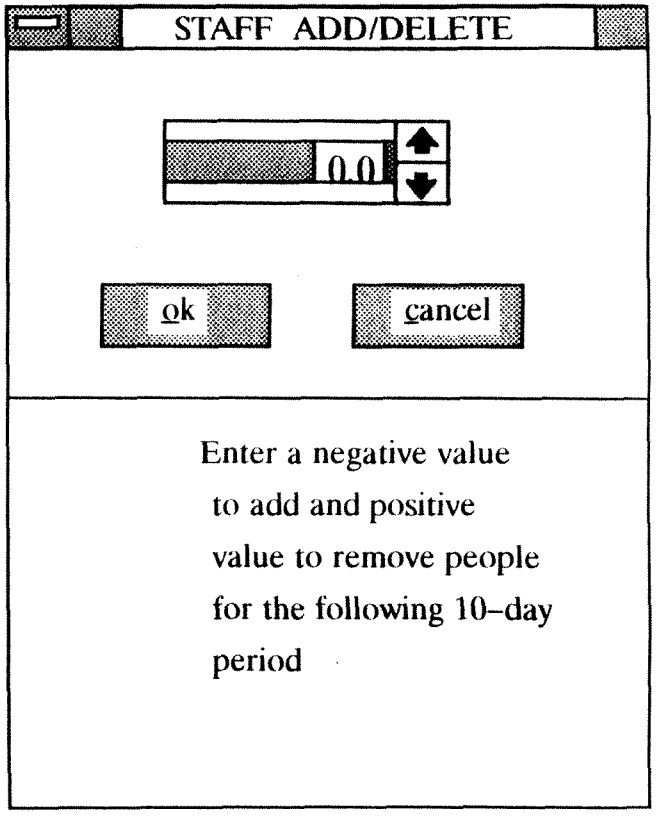

Figure VI.2.c

Figure VI.2. Decision Input Screens

Again, if player clicks on "ok" or use the "Return" keyboard button, a new window (Figure VI.2.c) appears for his/her last decision. If he/she enters a positive value it means he/she is either transferring some people from other departments or hiring some new people, and deleting staff works in the opposite way. When player is making his/her decisions about staff adding/deleting, he/she should consider some internal features of the model. First, when he/she decides to add new staff, they do not join the work force immediately, but rather gradually over a "hiring delay". Also, it takes some time for the new workers to become experienced (called assimilation delay). This is important because experienced workers perform with twice the efficiency of the new ones. Finally, there is an internal quitting rate which is not under player's control. Some people may quit in the middle of the project and he/she may have no control over it. This situation is handled by the model. Therefore when he/she is adding or removing some work force he/she should take these features into consideration.

If player clicks "cancel" button on any decision window, he/she returns to the main game window. If he/she clicks on "ok" on the last decision window, the game starts. It runs for some period of time (10/20 days for small/large size project, respectively), at the end of which it stops and waits for his/her next decisions. Player should follow the procedure described above to enter new decisions. However, before he/she steps into making new decisions for the next time period he/she may want to obtain information about some of the variables involved.

Obtaining Information

The game contains an information system which allows player to monitor developments in all sectors of the project development. The

most important variables are on the main screen. Also, he/she can use "Analyze More Information" button to have a look at other selected variables (right button on the main screen shown in Fig. VI.1). As in many real situations, player is given a lot of information, a few of which are more useful than others (Figure VI.3.a). Player must try to select the most important and useful data to assist him/her in making his/her decisions. In order to analyze a variable (he/she can analyze one variable at a time), he/she should select one of the variables and than click "show info." or press "Return". Having done that, a graph showing the distribution of the selected variable up to the current time accompanied with an information window showing the current value of the concerned variable (with its unit) is displayed (Figure VI.3.b). If player clicks on "ok", it takes him/her back to Figure VI.3.a. Note that, the lines written in capital letters are sector headings, not variables, so if he/she happens to choose one of them, he/she gets an error message indicating that he/she should choose a variable. In order to go back to the main screen, he/she should click on "Back to Main".

End of the Game

There are four different ways in which the game may end. First, if player exceeds the time limit for the given project, the simulation stops and prompt with a dialogue box indicating his/her situation. In the second case, he/she may exceed the budget limit, and he/she sees a message indicating that he/she is out of budget. Third, which is the worst, player may be out of both budget and time, in this case again he/she gets a message indicating bankruptcy. The fourth and desired one is to finish it within the limits, in which case he/she gets a congratulation message.

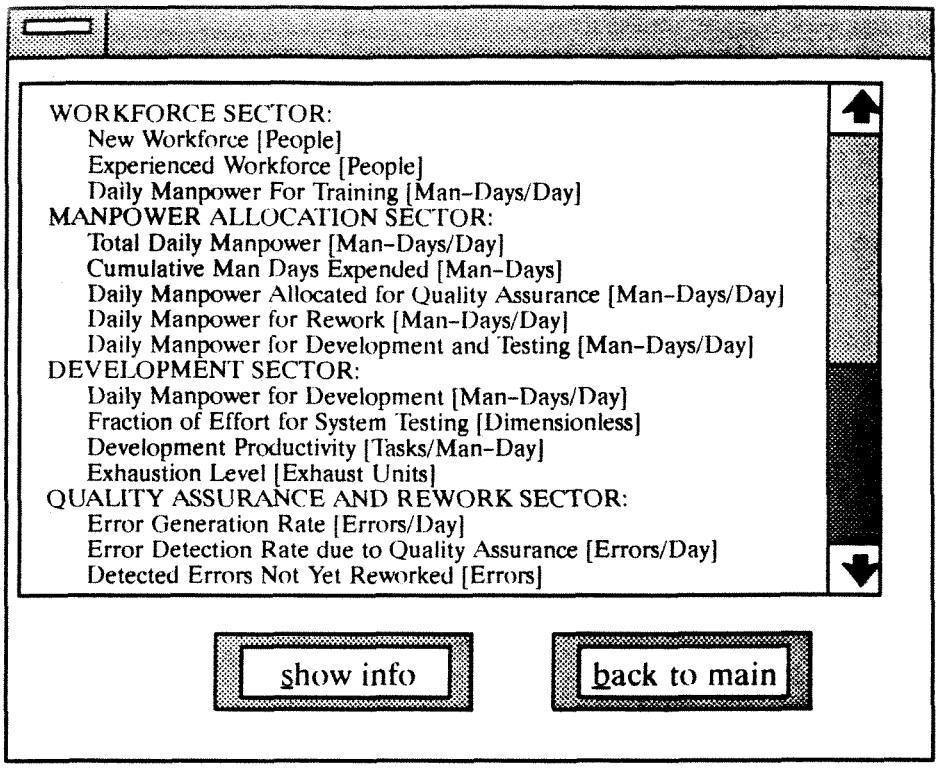

Figure VI.3.a

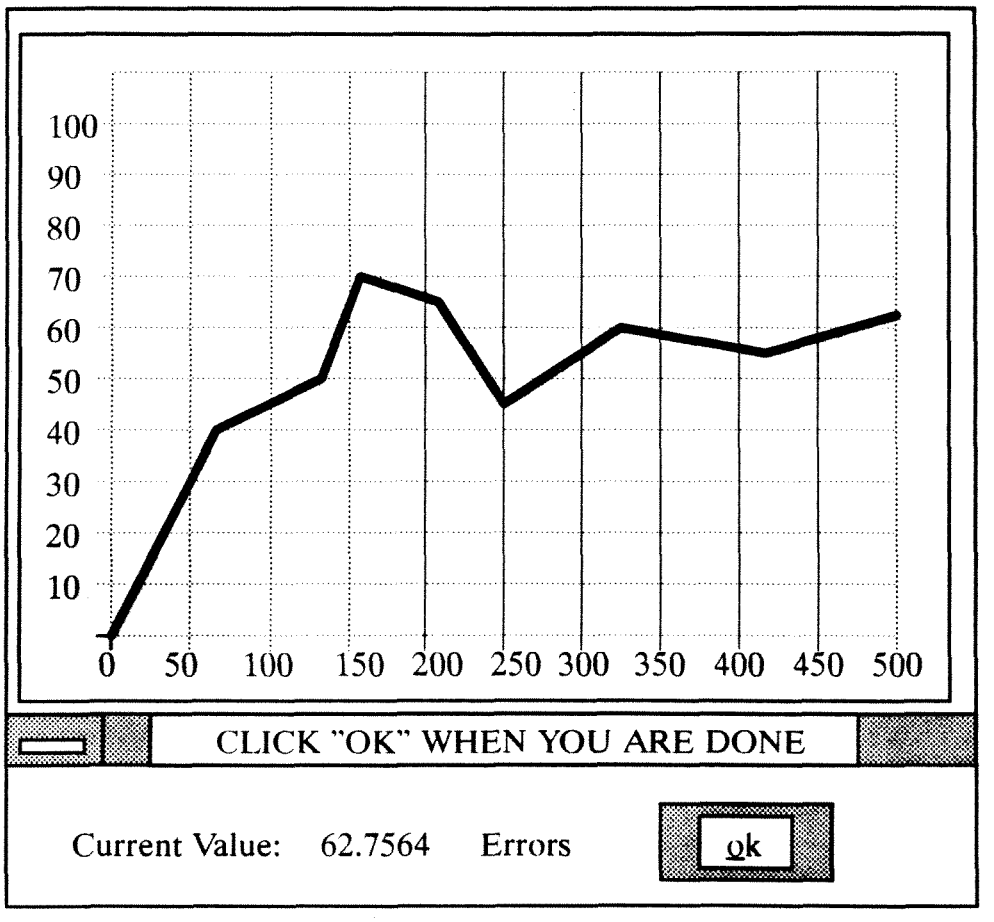

Figure VI.3.b

Figure VI.3. Information Screens

At the end of the game, player has the option of analyzing the variables both on the screen and by using "analyze more info." button.

If he/she would like to play again he/she may use the **Game** pull-down menu and select **New Game.** This takes him/her to the opening screen, where he/she can again select his/her options for the next game. If he/she would like to quit, simply select **Quit** from pull-down menu **Game.**

Options

Player can play the game with different combination of options which are specifically classified in three sections.

The first option is "skill: easy-difficult". In the easy case, delay accompanied with the addition of staff is equal to the decision period. Therefore when he/she decides to add staff at the beginning of the period, addition will be completed at the end of the same decision period. On other hand, in the difficult version, hiring delay is greater than the decision period. Thus he/she is not be able to get all of the staff he/she is trying to hire at the end of the period; some will join his/her staff in the next period.

The second option is "size: small-large". This option is related with the size of the project which is determined by **D**elivered **S**ource **I**nstructions (DSI). The size of the small project is 25,000 DSI and the size of the large one is 200,000. For the small size project the limits are 500 days and 2500 man-days, and for the large size project the limits are 1200 days and 30000 man-days. (Also note that, decision period for the small game is 10 days, and for the large game it is 20 days)

Finally, the third option is "Underestimation: w/o underestimation-w/ underestimation". Project size is estimated as number of tasks at the beginning of the game. If we know the exact size, and also if we have the guarantee that the size will remain the same, it means that there is no underestimation in the project size in terms of number of tasks. On the other hand, "w/ underestimation" indicates that there is an unknown amount of underestimation in project size, which makes the game more difficult.

VII. RESULTS AND CONCLUDING OBSERVATIONS

As a part of validation and testing phase, the game is exposed to criticisms by players. Eight players (2 faculty members, 4 graduate students and 2 undergraduate students) participated in playing the game, with the options of small project size, easy case, without underestimation. Performances of these players and the output of the model under the same conditions are given in Table VII.1. Players 2,4 and 8 were able to complete the project significantly earlier than the other players and the model, but, cumulative man days expended by all players were over 2,000. On the other hand, players 1,5,7 completed the project with less budget than other players, even though none of the players were able to finish the game with less budget expended than the model. Players 2,4 and 8 used more workforce in their project than the others did, which resulted in an increase in budget expended and decrease in scheduled completion date. Players 3,5 and 7 used less workforce than others, thus except player 3, they both ended up with low budget expended and high scheduled completion dates. Player 3's budget expenditure is higher than the other two's, but he was able to finish earlier. At the end of the project, all players allocated zero manpower to quality assurance and rework except 4,5 and 8.

Player #	1	2	3	4	5	6	7	8	Model
TIME	262	162	232	161	295	287	266	188	271
SCHCDT (Scheduled Completion Date)	261.77	161.54	231.21	160.41	294.02	286.96	265.75	187.28	270.3
JBSZMD (Total Job Size In Man Days)	1529.81	2039.70	2129.30	2267.36	1635.68	2486.81	1493.47	2055.40	1339.0
CUMMD (Cumulative Man-Days Expended)	1530.80	2045.44	2127.93	2276.02	1640.11	2487.98	1494.05	2060.52	1337.3
PJBSZ (Currently Perceived Job Size)	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.6
CMTKDV (Cumulative Tasks Developed)	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.6
DMPDVT (Daily Manpower For Developemnt/Testing)	4.84	9.30	1.22	10.11	2.85	5.55	2.17	6.88	4.5
DMPQA (Daily Manpower For QA)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.00
DMPRW (Daily Manpower For Rework)	0.00	0.00	0.00	3.37	1.90	0.00	0.00	1.21	0.00
WFNEW (New Workforce)	0.01	0.00	0.00	0.03	0.10	0.00	0.00	0.00	0.21
WFEXP (Experienced Workforce)	4.83	9.30	1.22	13.45	4.65	5.55	2.17	8.52	4.53

Table VII.1. Comparison of performances of 8 players at the end of the game and the simulation model's performance

The distribution of some of the variables throughout three example game sessions are shown in Figures VII.1.a, VII.1.b, VII.2.a, VII.2.b, VII.3.a & VII.3.b. The behavior of project size in terms of number of tasks are expectedly the same in all of the outputs. Also, the behavior of the cumulative tasks developed is similar in all of the outputs. At the end, cumulative tasks developed is equal to project size in terms of number of tasks. In all of the outputs, at the end of the project cumulative man days expended is equal to project size in man days and cumulative tasks developed is equal to project size in terms of number of tasks. The behavior of the cumulative man days expended is similar in all of the outputs, it starts from zero, and increases till it reaches job size in man days at the end of project (Fig. IV.2.a, VII.1.a, VII.2.a & VII.3.a).

On the other hand, there is a slight difference in job size in man days in different outputs. Even though the behavior is basically similar, in two of the outputs there is an early increase (Fig. VII.2.a & VII.3.a) which can be explained by having high level of total workforce in the project. In Fig. VII.1.a, we can not the see the same increase due to relatively small workforce level. In the model simulation, job size in man days stays constant because of it's using optimum parameters for workforce, schedule adjustments and manpower allocation.

One of the significant differences between the behavior patterns obtained from the interactive game and the outputs of the simulation is the behavior of the scheduled completion date. In the model, scheduled completion date stays constant for a while, and then there is an upward adjustment (Fig. IV.2.a). On the other hand, in the interactive games, there is a decline at the beginning, which is a result of allocating more manpower to

a tendency for it to go down at the early stages of the project, then it starts going up. Even though the increase in scheduled completion date is smooth in the model, it usually displays a sharper increase in the games. The last, but not the least, significant difference between the model and the games is the behavioral pattern of manpower allocation for different sectors. The model shows a smooth allocation of manpower to development, quality assurance and rework at the early stages of the project. However, in the games we see sharp fluctuations in the patterns.

These are actually not the only variables that are worth examining. The model consists of tens of variables that have potentials for further examination. Further extensive research may be done to analyze more variables statistically, and search for relationships. More data can be gathered and behavioral responses of the players can be examined.

This research can also be extended in terms of improving certain features of the game. Currently, the game runs by using WingZ which runs under Windows 3.0. Since it uses two environments simultaneously, it lacks efficiency in terms of speed. It can be made faster by installing it under OS/2 instead of Windows 3.0. Another solution would be to program the game directly under Windows 3.0.

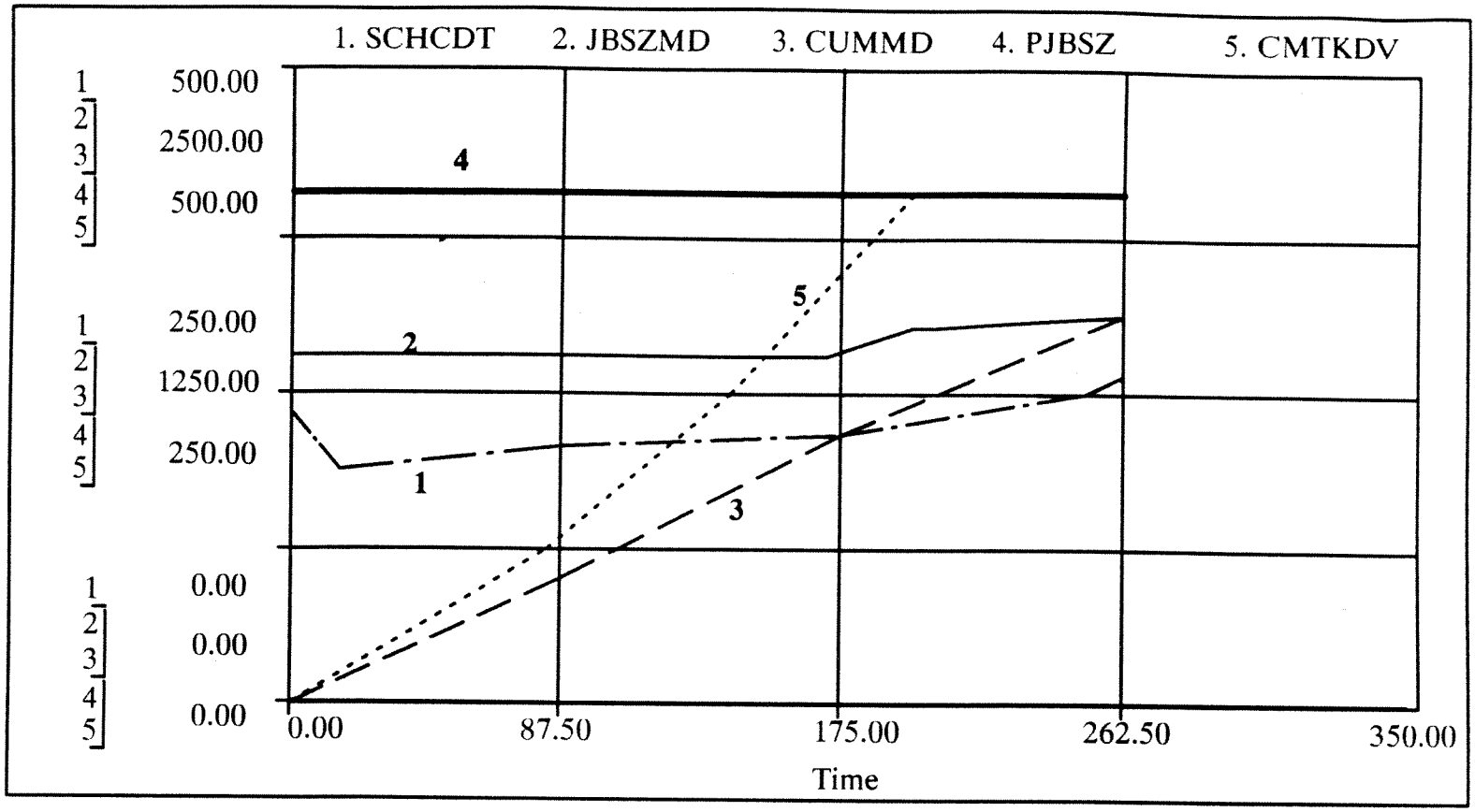

Figure VII.1.a

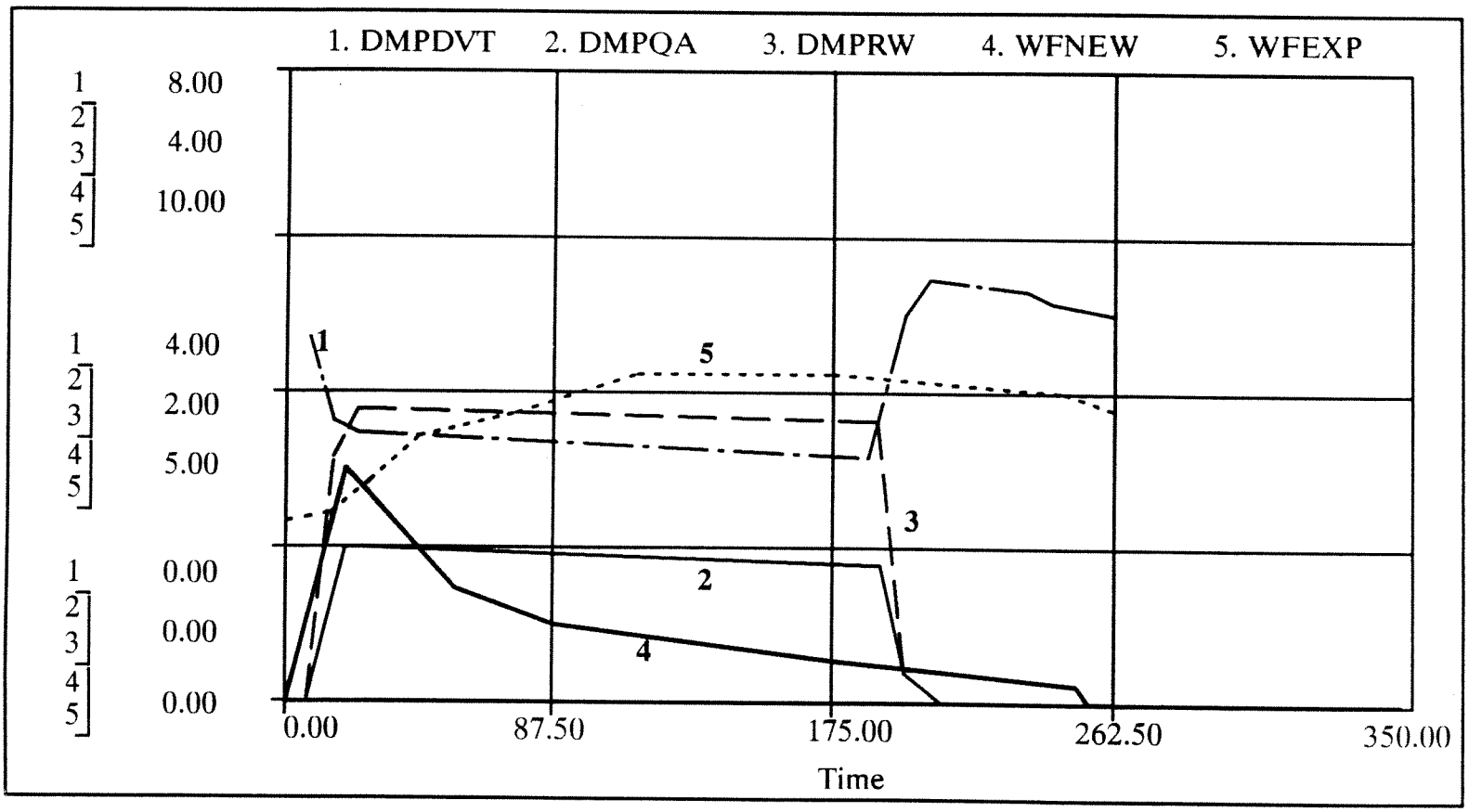

Figure VII.1.b

Figure VII.1. Performance of Player # 1

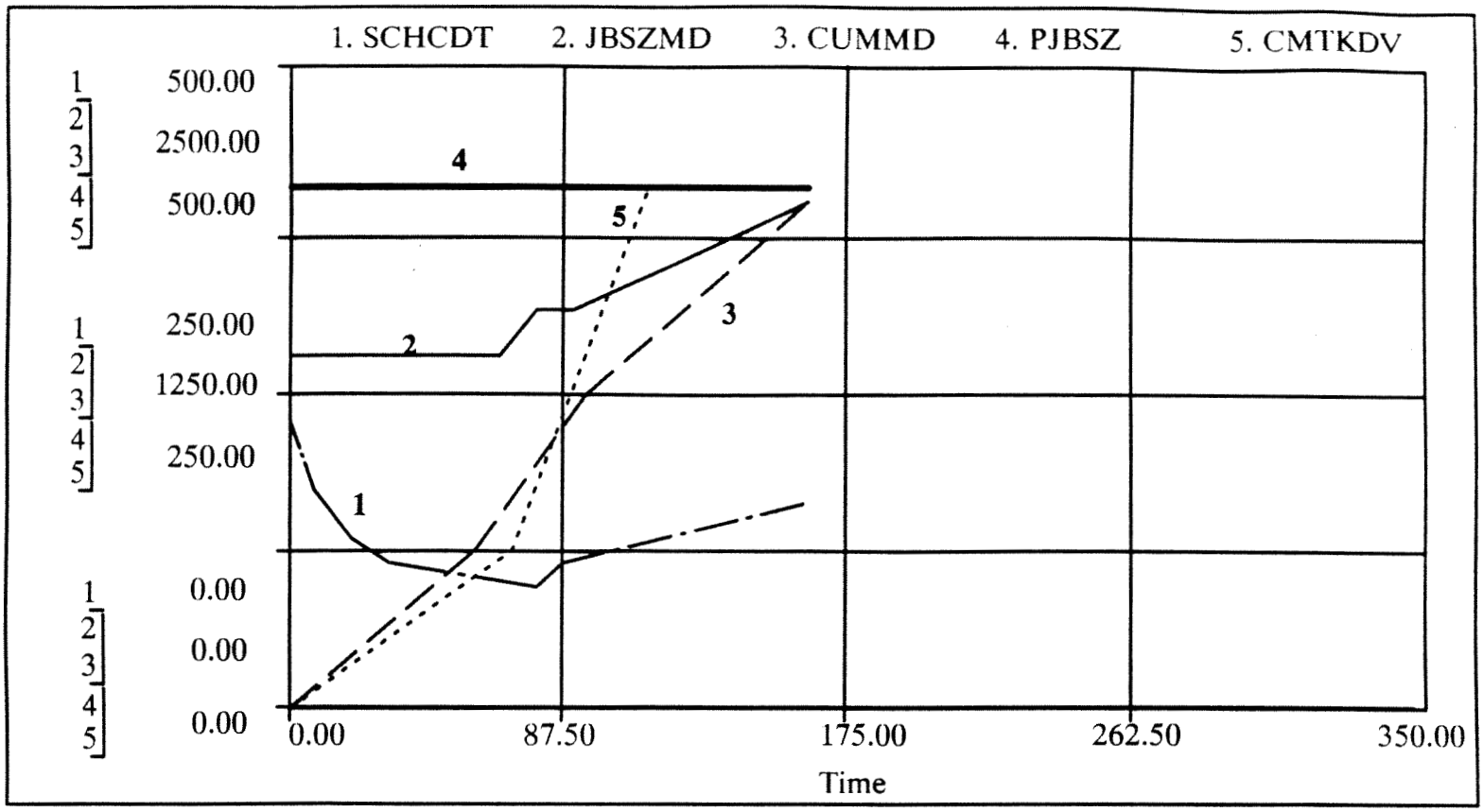

Figure VII.2.a

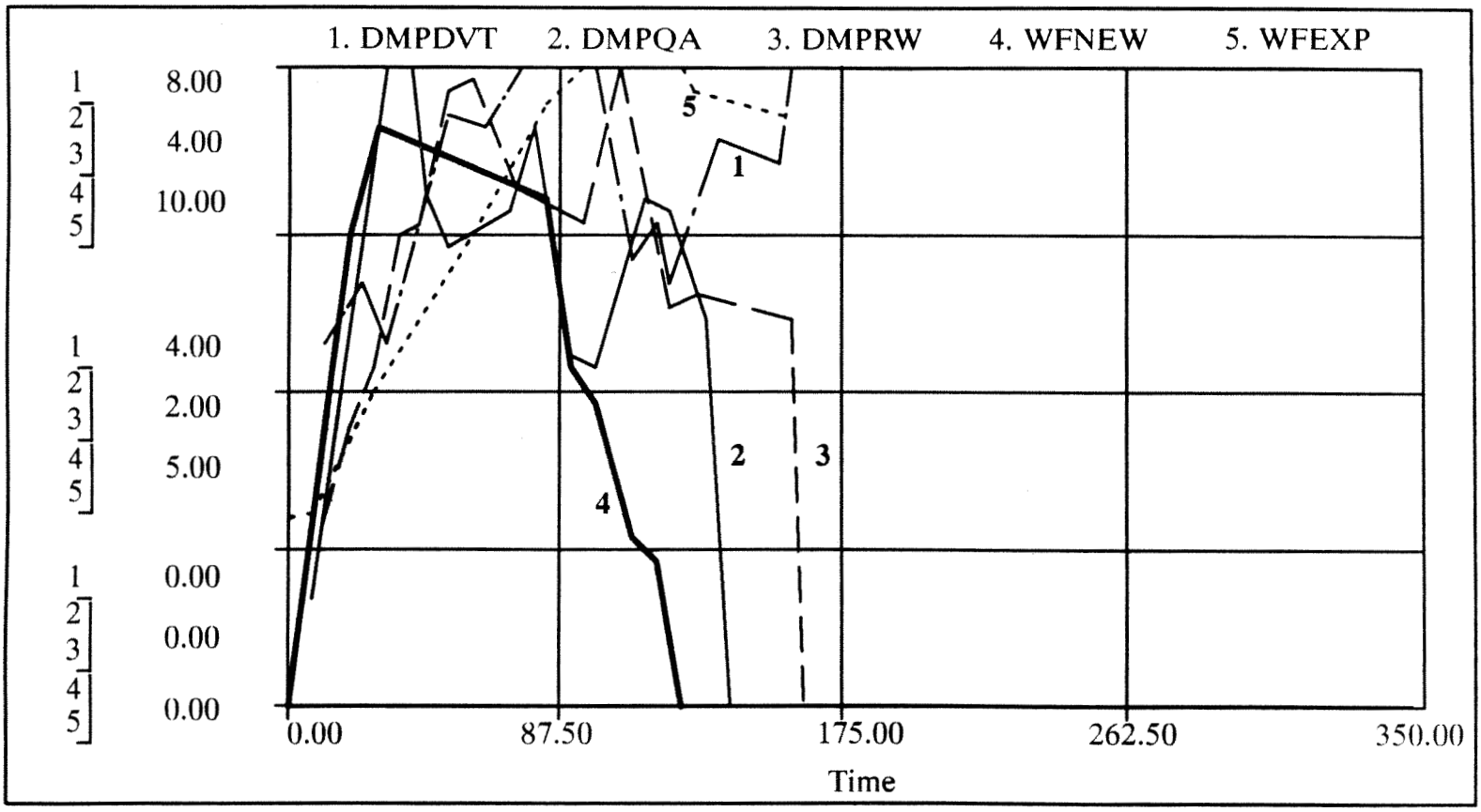

Figure VII.2.b

Figure VII.2. Performance of Player # 2

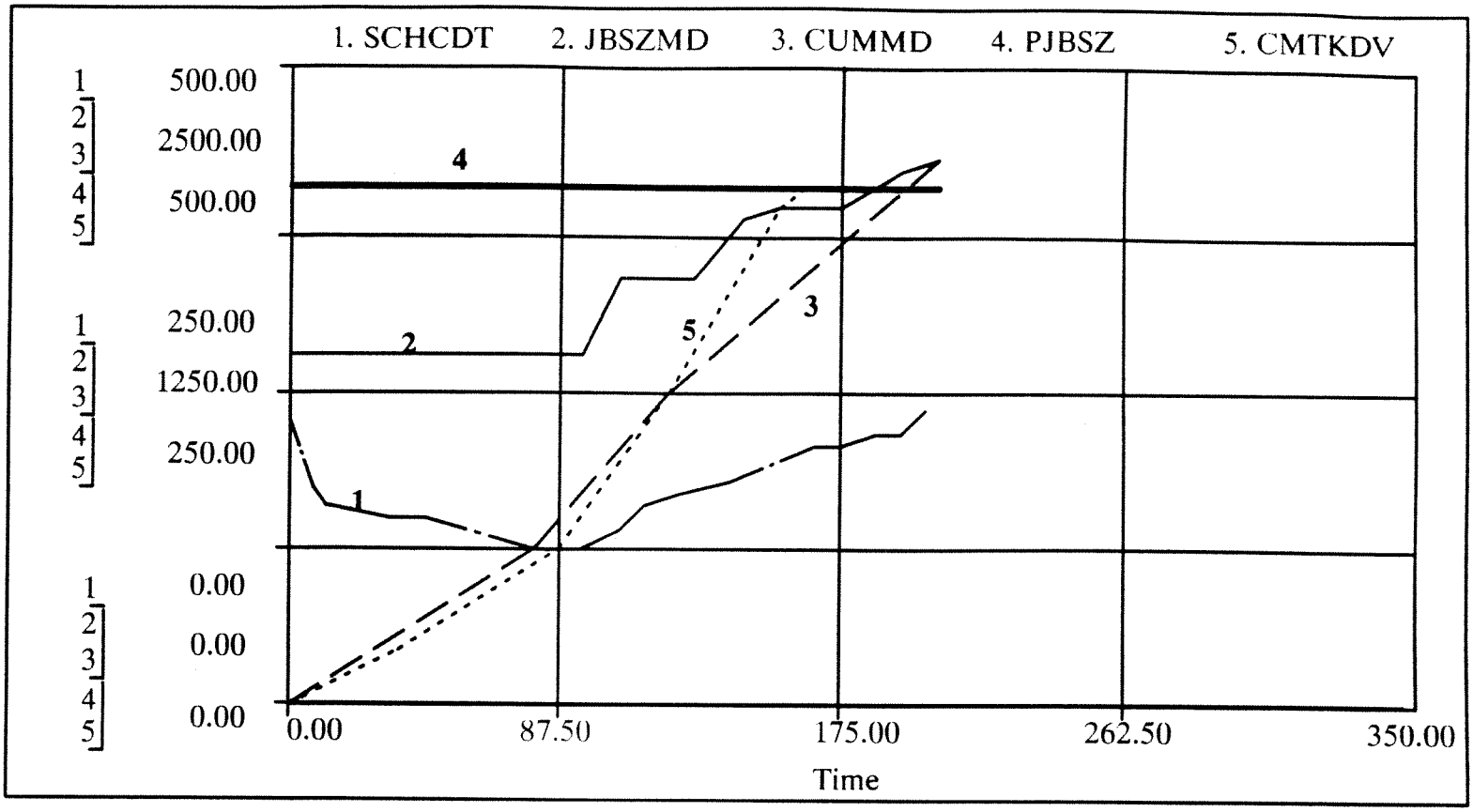

Figure VII.3.a

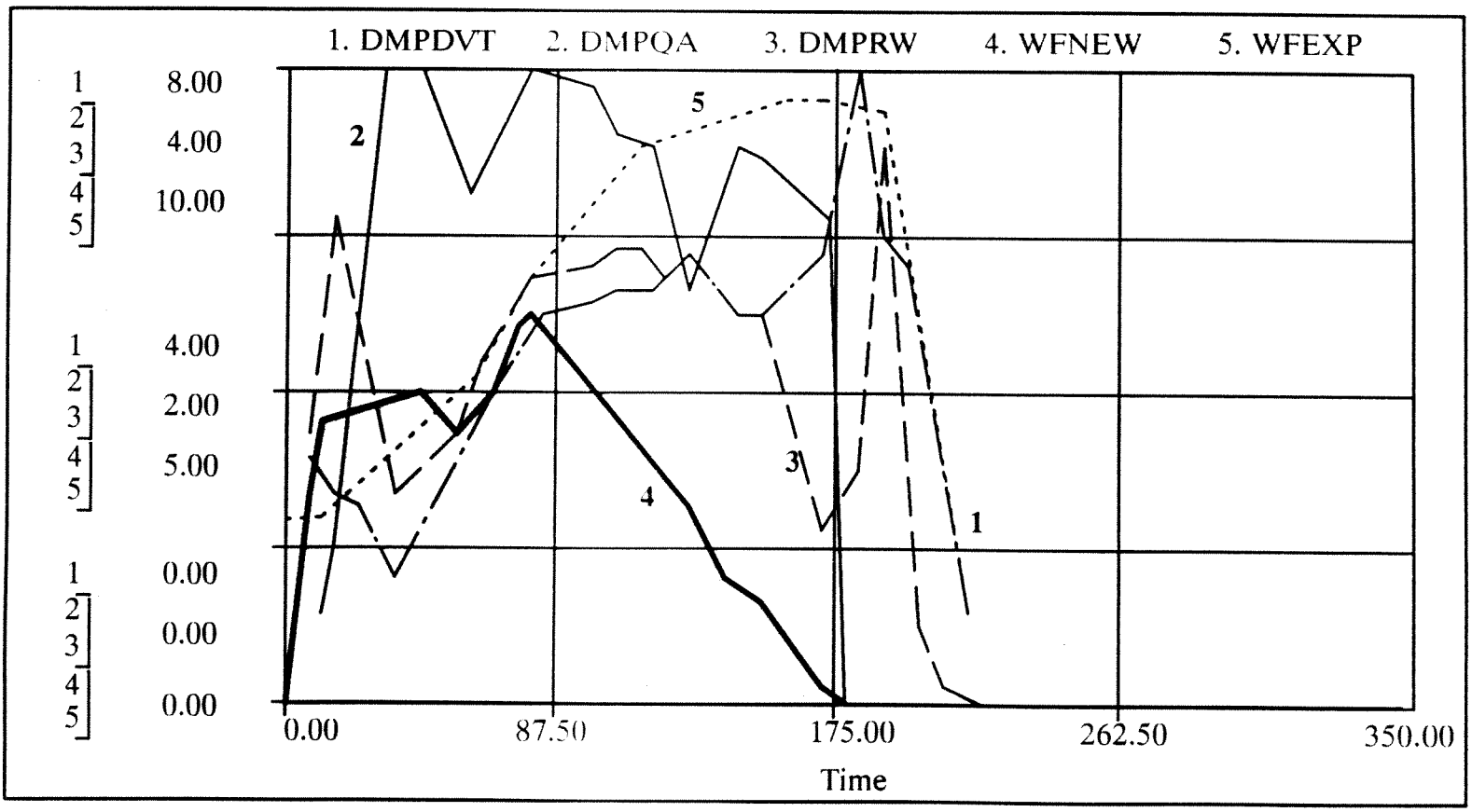

Figure VII.3.b

Figure VII.3. Performance of Player # 3

VII. REFERENCES

1. Tarek K. Abdel Hamid and Stuart E. Madnick. 1991. Software Project Dynamics, An Integrated Approach. Prentice Hall, Englewood Cliffs, New Jersey.

2. Kim, Daniel H. 1989. Learning Laboratories : Designing a Reflective Learning Environment. In Computer-Based Management of Complex Systems, eds. Peter Milling and Erich O.K. Zahn, Springer-Verlag, Berlin.

3. Tarek K. Abdel-Hamid and Stuart E. Madnick. Lessons Learned from Modeling the Dynamics of Software Development, Communication of the ACM - Dec. 1989 v2 n12.

4. Tarek K. Abdel-Hamid and Stuart E. Madnick. Software Productivity: Potential, Actual and Perceived, System Dynamics Review Summer 1989 v5 n2.

5. Graham, Alan K.; Senge, Peter M.; Sterman,John D.; Morecroft, John D.W. 1989. Computer-Based Case Studies in Management Education and Research. In Computer-Based Management of Complex Systems, eds. Peter Milling and Erich O.K. Zahn, Springer-Verlag, Berlin.

6. Meadows, Dennis. 1989. Gaming to Implement System Dynamics Models. In Computer-Based Management of Complex Systems, eds. Peter Milling and Erich O.K. Zahn, Springer-Verlag, Berlin.

7. Daniel, H. Kim. Sun MicroSystems, MIT Sloan School of Management Working Paper - Dec. 1988.

8. Diehl, Stanford. Windows Takes on WingZ, Byte Nov. 1990 v15 n12 p221(3).

9. High Performance WingZ, Which Computer? Oct.90 v13 n10 p84(1).

10. Barlas, Yaman. Multiple Tests for Validation of System Dynamics Type of Simulation Models. In European Journal of Operations Research - 1989 vol.42 no.1 pp.59-87.

11. Stella II, User's Guide. High Performance Systems. Hanover, New Hampshire, Dec. 1990.

APPENDICES

1. Model Equations in Stella

2. Model Equations in WingZ

```
AFMDPJ(t) = AFMDPJ(t - dt) + (WRADJR) * dt
  INIT AFMDPJ = NFMDPJ
  INFLOWS:
    WRADJR = (WKRTS-AFMDPJ)/WKRADY
BRKDTM(t) = BRKDTM(t - dt) + (BREAKDOWN) * dt
  INIT BRKDTM = -1
  INFLOWS:
    BREAKDOWN = (MAX(BRKDTM,SW)-BRKDTM)/DT
CMTKDV(t) = CMTKDV(t - dt) + (SDVRT2) * dt
  INIT CMTKDV = 0
  INFLOWS:
    SDVRT2 = SDVRT1
CMTSMD(t) = CMTSMD(t - dt) + (DMPTST) * dt
  INIT CMTSMD = 0
  INFLOWS:
    DMPTST = DMPDVT*FREFTS
CUMMD(t) = CUMMD(t - dt) + (TOTDMP) * dt
  INIT CUMMD = 0.0001
  INFLOWS:
    TOTDMP = TOTWF*ADMPPS
CUMTKT(t) = CUMTKT(t - dt) + (TSRATE) * dt
  INIT CUMTKT = 0
  INFLOWS:
    TSRATE = MIN(CUMTQA/DT,DMPTST/TMPNPT)
CUMTQA(t) = CUMTQA(t - dt) + (QART2 - TSRATE) * dt
  INIT CUMTQA = 0
  INFLOWS:
    QART2 = TSKWK2/AQADLY
  OUTFLOWS:
    TSRATE = MIN(CUMTQA/DT,DMPTST/TMPNPT)
DISC1(t) = DISC1(t - dt) + (RTDSTK - RTINCT1) * dt
  INIT DISC1 = 0
  INFLOWS:
    RTDSTK = UNDJTK*PUTDPD/100
  OUTFLOWS:
    RTINCT1 = DISC1/DLINCT
DISC2(t) = DISC2(t - dt) + (RTINCT1 - RTINCT2) * dt
  INIT DISC2 = 0
  INFLOWS:
    RTINCT1 = DISC1/DLINCT
  OUTFLOWS:
    RTINCT2 = DISC2/DLINCT
DTCERR(t) = DTCERR(t - dt) + (ERRDRT - RWRATE) * dt
  INIT DTCERR = 0
  INFLOWS:
```

ERRDRT = PTDTER*QADJUST

OUTFLOWS:

RWRATE = DMPRW/RWMPPE

EXHLEV(t) = EXHLEV(t - dt) + (RIEXHL - RDEXHL) * dt

INIT EXHLEV = 0

INFLOWS:

RIEXHL = GRAPH((1-AFMDPJ)/(1-NFMDPJ))
(-0.5, 2.5) (-0.4, 2.2) (-0.3, 1.9) (-0.2, 1.6) (-0.1, 1.3) (0, 1.15) (0.1, 0.9) (0.2, 0.8) (0.3, 0.7) (0.4, 0.6) (0.5, 0.5) (0.6, 0.4) (0.7, 0.3) (0.8, 0.2) (0.9, 0) (1, 0)

OUTFLOWS:

RDEXHL = IF RIEXHL > 0 THEN 0 ELSE EXHLEV/EXHDDY

JBSZMD(t) = JBSZMD(t - dt) + (IRDVDT + IRTSDT1 + ARTJBM) * dt

INIT JBSZMD = DEVMD+TSTMD

INFLOWS:

IRDVDT = (RTINCT2/ASSPRD)*FADHWO

IRTSDT1 = (RTINCT2/PRTPRD)*FADHWO

ARTJBM = (MDRPTN+CUMMD-JBSZMD)/DAJBMD

PJBSZ(t) = PJBSZ(t - dt) + (RTINCT2) * dt

INIT PJBSZ = PJBDSI/DSIPTK

INFLOWS:

RTINCT2 = DISC2/DLINCT

PRTPRD(t) = PRTPRD(t - dt) + (DUMMYRATE) * dt

INIT PRTPRD = SELECT

INFLOWS:

DUMMYRATE = (SELECT-PRTPRD)/TSTSPD

PRWMPE(t) = PRWMPE(t - dt) + (CHANGE1) * dt

INIT PRWMPE = 0.5

INFLOWS:

CHANGE1 = (RWMPPE-PRWMPE)/TARMPE

PTDTER(t) = PTDTER(t - dt) + (ERRGRT - ERRSRT - ERRDRT) * dt

INIT PTDTER = 0

INFLOWS:

ERRGRT = SDVRT1*ERRPTK

OUTFLOWS:

ERRSRT = .1*PTDTER

ERRDRT = PTDTER*QADJUST

RLXTMC(t) = RLXTMC(t - dt) + (DEEXHAUST - DISCHARGE) * dt

INIT RLXTMC = 0

INFLOWS:

DEEXHAUST = IF (EXHLEV/MXEXHT) >= 0.1 THEN 1 ELSE (-RLXTMC/DT)

OUTFLOWS:

DISCHARGE = IF OVWDTH = 0 THEN (RLXTMC/DT) ELSE 0

SCHCDT(t) = SCHCDT(t - dt) + (CHANGE3) * dt

INIT SCHCDT = TDEV

INFLOWS:
  CHANGE3 = (INDCDT-SCHCDT)/SCHADT
TSKWK1(t) = TSKWK1(t - dt) + (SDVRT1 - QART1) * dt
INIT TSKWK1 = 0
INFLOWS:
  SDVRT1 = MIN((DMPSDV*SDVPRD),TSKPRM/DT)
OUTFLOWS:
  QART1 = TSKWK1/AQADLY
TSKWK2(t) = TSKWK2(t - dt) + (QART1 - QART2) * dt
INIT TSKWK2 = 0
INFLOWS:
  QART1 = TSKWK1/AQADLY
OUTFLOWS:
  QART2 = TSKWK2/AQADLY
TSZZMD(t) = TSZZMD(t - dt) + (IRTSDT2 + TSZRATE) * dt
INIT TSZZMD = TSTMD

INFLOWS:
  IRTSDT2 = IRTSDT1
  TSZRATE = IF FREFTS >= 0.9 THEN (ARTJBM/DT) ELSE 0
UNDERR(t) = UNDERR(t - dt) + (ERRSRT + RGNRT - CORRECT) * dt
INIT UNDERR = 0
INFLOWS:
  ERRSRT = .1*PTDTER
  RGNRT = SDVRT1*SMTERRDSY
OUTFLOWS:
  CORRECT = MIN(TSRATE*UNDERRDSY,UNDERR/DT)
UNDJTK(t) = UNDJTK(t - dt) + (- RTDSTK) * dt
INIT UNDJTK = RJBSZ-PJBSZ
OUTFLOWS:
  RTDSTK = UNDJTK*PUTDPD/100
WFEXP(t) = WFEXP(t - dt) + (ASIMRT - QUITRT - EXPTRR) * dt
INIT WFEXP = WFSTRT
INFLOWS:
  ASIMRT = WFNEW/ASIMDY
OUTFLOWS:
  QUITRT = WFEXP/AVEMPT
  EXPTRR = MIN(WFEXP/DT,TRNFRT-NEWTRR)
WFNEW(t) = WFNEW(t - dt) + (HIRERT - ASIMRT - NEWTRR) * dt
INIT WFNEW = 0
INFLOWS:
  HIRERT = MAX(0,WFGAP/HIREDY)
OUTFLOWS:
  ASIMRT = WFNEW/ASIMDY
  NEWTRR = MIN(TRNFRT,WFNEW/DT)

) ACTSPD = CUMTKT/(CMTSMD+.001)
) ADMPPS = 1
) AFMPQA = PFMPQA*(1+ADJQA)
) ANERPT = MAX(PTDTER/(TSKWK+.0001),0)
) ANPPRD = FRWFEX*NPWPEX+(1-FRWFEX)*NPWPNE
) AQADLY = 5
) ASIMDY = 80
) ASSPRD = PJDPRD*WTPJDP+PRDPRD*(1-WTPJDP)
) AVEMPT = 673
) CELNWH = FTEXWF*MNHPXS
) CELTWF = CELNWH+WFEXP
) CTRLSW = 1
) DESECR = DTCERR/DESRWD
) DESRWD = 15
) DEVMD = DEVPRT*TOTMD
) DEVPRT = 0.8
) DLINCT = 5
) DMPATR = TOTDMP-DMPTRN
) DMPDVT = DMPSWP-DMPRW
) DMPQA = MIN((AFMPQA*TOTDMP),.3*DMPATR)
) DMPRW = MIN((DESECR*PRWMPE),DMPSWP)
) DMPSDV = DMPDVT*(1-FREFTS)
) DMPSWP = DMPATR-DMPQA
) DMPTRN = WFNEW*TRPNHR
) DSIPTK = 60
) ERRDSY = ANERPT*1000/DSIPTK
) ERRPTK = NERPTK*MERGSP*MERGWM
) EWKRTS = IF WKRTS >= AFMDPJ THEN 1 ELSE 0.75
) EXHDDY = 20
) EXSABS = MAX(0,(TEXABS*MDRM-TMDPSN))
) FRWFEX = WFEXP/TOTWF
) FTEQWF = TOTWF*ADMPPS
) FTEXWF = WFEXP*ADMPPS
) HIREDY = 40
) INDCDT = TIME+TIMEPR
) INUDST = .5
) MAXMHR = 1
) MAXSHR = (OVWDTH*FTEQWF*MAXMHR)*WTOVWK
) MDHDL = IF PMDSHR >= 0 THEN (MIN(MAXSHR,PMDSHR)*CTRLSW)
                ELSE ((-EXSABS)*CTRLSW)
) MDPNNT = TSKPRM/ASSPRD
) MDPNRW = DTCERR*PRWMPE
) MDPNTS = TSTPRM/PRTPRD
) MDPRNT = MAX(0,MDRM-MDPNRW-MDPNTS)
) MDRM = MAX(.0001,JBSZMD-CUMMD)

) MDPPTN = MDRM+SHRRPT
) MDSWCH = 1
) MNHPXS = 7
) MPDMCL = AFMDPJ*(1-COMMOH)
) MSZTWO = .01
) MXEXHT = 50
) NERPTK = NERPK*DSIPTK/1000
) NFMDPJ = 0.6
) NPWPEX = 1
) NPWPNE = 0.5
) OVWDTH = NOVWDT*MODTEX
) PBWKRS = IF PMDSHR >= 0 THEN (MDHDL/(FTEQWF*(OVWDTH+.0001)))
ELSE (MDHDL/(TMDPSN-MDHDL+.0001))
) PJBAWK = CMTKDV/PJBSZ
) PJBDSI = RJBDSI*(1-UNDEST)
) PJBPWK = (CMTKDV/PJBSZ)*100
) PJDPRD = TSKPRM/(MDPRNT+0.1)
) PLTSPD = PJBSZ/TSZZMD
) PMDSHR = TMDPSN-MDRM
) POTPRD = ANPPRD*MPPTPD
) PRDPRD = CMTKDV/(CUMMD-CMTSMD)
) PSZDCT = TKDSCV/ASSPRD
) PTKTST = CUMTKT/PJBSZ
) RJBDSI = 25000
) RJBSZ = RJBDSI/DSIPTK
) RSZDCT = PSZDCT/(MDPRNT+.0001)
) RWMPPE = NRWMPE/MPDMCL
) SCHCOM = 1
) SCHPR = (TMDPSN-MDRM)/MDRM
) SCSWCH = 1
) SDVPRD = POTPRD*MPDMCL
) SELECT = IF CUMTKT > 0 THEN ACTSPD ELSE PLTSPD
) SHRRPT = PMDSHR-MDHDL
) SMTERRDSY = SMTH1(UNDERRDSY,TSAEDS)
) SW = IF OVWDTH = 0 THEN (TIME+DT) ELSE 0
) TARMPE = 10
) TDEV = SCSWCH*((19*2.5*EXP(0.38*LOGN(TOTMD/19)))*SCHCOM)+
(1-SCSWCH)*TDEV1
) TDEV1 = 0
) TEAMSZ = (TOTMD/TDEV)/ADMPPS
) TIMEPR = MDRM/(WFS*ADMPPS)
) TIMERM = MAX(SCHCDT-TIME,0)
) TKDSCV = DISC1+DISC2
) TMDPSN = MDPNNT+MDPNTS+MDPNRW
) TMPNPE = .25

```
TMPNPT = (TSTOVH*DSIPTK/1000+TMPNPE*UNDERRDSY)/MPDMCL
TOTMD = MDSWCH*(((2.4*EXP(1.05*LOGN(PJBDSI/1000)))*19)*(1-UNDESM))+
(1-MDSWCH)*TOTMD1
TOTMD1 = 0
TOTWF = WFNEW+WFEXP
TRNFRT = MAX(0,-WFGAP/TRNSDY)
TRNSDY = 10
TRPNHR = 0.2
TSAEDS = 40
TSKPRM = PJBSZ-CMTKDV
TSKWK = TSKWK1+TSKWK2
TSTMD = (1-DEVPRT)*TOTMD
TSTOVH = 1
TSTPRM = PJBSZ-CUMTKT
TSTSPD = 50
UNDERRDSY = UNDERR/(CUMTQA+.1)
UNDESM = 0
UNDEST = 0
WFGAP = WFS-TOTWF
WFINDC = (MDRM/(TIMERM+.001))/ADMPPS
WFNEED = MIN((WCWF*WFINDC+(1-WCWF)*TOTWF),WFINDC)
WFS = MIN(CELTWF,WFNEED)
WFSTRT = TEAMSZ*INUDST
WKRADY = NWRADY*EWKRTS
WKRTS = (1+PBWKRS)*NFMDPJ
WTOVWK = IF TIME >= (BRKDTM+RLXTMC) THEN 1 ELSE 0
WTPJDP = MPWDEV*MPWREX
ADJQA = GRAPH(SCHPR)
(0.00, 0.00), (0.1, -0.025), (0.2, -0.15), (0.3, -0.35), (0.4, -0.475), (0.5, -0.5)
COMMOH = GRAPH(TOTWF)
(0.00, 0.00), (5.00, 0.015), (10.0, 0.06), (15.0, 0.135), (20.0, 0.24), (25.0, 0.375), (30.0, 0.54)
DAJBMD = GRAPH(TIMERM)
(0.00, 0.5), (20.0, 3.00)
FADHWO = GRAPH(RSZDCT/(MSZTWO+.001))
(0.00, 0.00), (0.2, 0.00), (0.4, 0.00), (0.6, 0.00), (0.8, 0.00), (1.00, 0.00), (1.20, 0.7), (1.40, 0.9),
(1.60, 0.975), (1.80, 1.00), (2.00, 1.00)
FREFTS = GRAPH(TSKPRM/PJBSZ)
(0.00, 1.00), (0.04, 0.5), (0.08, 0.28), (0.12, 0.15), (0.16, 0.05), (0.2, 0.00)
MERGSP = GRAPH(SCHPR)
(-0.4, 0.9), (-0.2, 0.94), (0.00, 1.00), (0.2, 1.05), (0.4, 1.14), (0.6, 1.24), (0.8, 1.36), (1.00, 1.50)
MERGWM = GRAPH(FRWFEX)
(0.00, 2.00), (0.2, 1.80), (0.4, 1.60), (0.6, 1.40), (0.8, 1.20), (1.00, 1.00)
MODTEX = GRAPH(EXHLEV/MXEXHT)
(0.00, 1.00), (0.1, 0.9), (0.2, 0.8), (0.3, 0.7), (0.4, 0.6), (0.5, 0.5), (0.6, 0.4), (0.7, 0.3), (0.8, 0.2),
(0.9, 0.1), (1.00, 0.00)
```

} MPPTPD = GRAPH(PJBAWK)
(0.00, 1.00), (0.1, 1.01), (0.2, 1.03), (0.3, 1.05), (0.4, 1.09), (0.5, 1.15), (0.6, 1.20), (0.7, 1.22), (0.8, 1.25), (0.9, 1.25), (1.00, 1.25)
} MPWDEV = GRAPH(PJBPWK/100)
(0.00, 1.00), (0.1, 1.00), (0.2, 1.00), (0.3, 1.00), (0.4, 1.00), (0.5, 1.00), (0.6, 0.975), (0.7, 0.9), (0.8, 0.75), (0.9, 0.5), (1.00, 0.00)
} MPWREX = GRAPH((1-MDPRNT/(JBSZMD-TSZZMD)))
(0.00, 1.00), (0.1, 1.00), (0.2, 1.00), (0.3, 1.00), (0.4, 1.00), (0.5, 1.00), (0.6, 0.975), (0.7, 0.9), (0.8, 0.75), (0.9, 0.5), (1.00, 0.00)
} NERPK = GRAPH(PJBAWK)
(0.00, 25.0), (0.2, 23.9), (0.4, 21.6), (0.6, 15.9), (0.8, 13.6), (1.00, 12.5)
} NOVWDT = GRAPH(TIMERM)
(0.00, 0.00), (10.0, 10.0), (20.0, 20.0), (30.0, 30.0), (40.0, 40.0), (50.0, 50.0)
} NRWMPE = GRAPH(PJBAWK)
(0.00, 0.4), (0.2, 0.366), (0.4, 0.32), (0.6, 0.27), (0.8, 0.22), (1.00, 0.15)
} NWRADY = GRAPH(TIMERM)
(0.00, 2.00), (5.00, 3.50), (10.0, 5.00), (15.0, 6.50), (20.0, 8.00), (25.0, 9.50), (30.0, 10.0)
} PFMPQA = GRAPH(PJBAWK)
(0.00, 0.15), (0.1, 0.15), (0.2, 0.15), (0.3, 0.15), (0.4, 0.15), (0.5, 0.15), (0.6, 0.15), (0.7, 0.15), (0.8, 0.15), (0.9, 0.15), (1.00, 0.00)
} PUTDPD = GRAPH(PJBPWK)
(0.00, 0.00), (20.0, 0.4), (40.0, 2.50), (60.0, 5.00), (80.0, 10.0), (100, 100)
} QADJUST = GRAPH(AFMPQA)
(0.00, 0.005), (0.1, 0.325), (0.2, 0.555), (0.3, 0.69), (0.4, 0.795), (0.5, 0.85), (0.6, 0.9), (0.7, 0.905), (0.8, 0.905), (0.9, 0.905), (1.00, 0.905)
} SCHADT = GRAPH(TIMERM)
(0.00, 0.5), (5.00, 10.0)
} TEXABS = GRAPH(TMDPSN/MDRM)
(0.00, 0.00), (0.1, 0.2), (0.2, 0.4), (0.3, 0.55), (0.4, 0.7), (0.5, 0.8), (0.6, 0.9), (0.7, 0.95), (0.8, 1.00), (0.9, 1.00), (1.00, 1.00)
} WCWF = GRAPH(TIMERM/(HIREDY+ASIMDY))
(0.00, 0.00), (0.3, 0.00), (0.6, 0.1), (0.9, 0.4), (1.20, 0.85), (1.50, 1.00), (1.80, 1.00), (2.10, 1.00), (2.40, 1.00), (2.70, 1.00), (3.00, 1.00)

App. 2

```
(   AUXILIARY VARIABLES & RATES & STOCKS   )
define period,loop,cont,r

r = rtemp

loop = 1

put wfexp+wfnew into totwf
put totwf*admpps into totdmp


      put wfnew*trpnhr into (dmptrn)i4
      put totdmp-(dmptrn)i4 into (dmpatr)i5

      run script "c:\wingz\afmpqa.scz"
      if i1 = 1
            put min((y1*totdmp),0.9*(dmpatr)i5) into (dmpqa)i6
            put (dmpatr-dmpqa) i5-i6 into (dmpswp)i7
            run script "c:\wingz\dmprw.scz"
            if i2 = 1
                  run script "c:\wingz\wf.scz"
                  if i3 = 1
                        put y1 into afmpqa
                        put y2 into perdmprw
                        put y3 into wf

                        put y1*100 into y99
                        put y2*100 into y98
                        put y3 into y97

                        if hiredy > inputperiod
                           put (wfgap+wf-(i8(temphirert)*inputperiod)+
                                ((i12(tempnewtrr)*inputperiod)+
                                (i13(tempexptrr)*inputperiod)) into wfgap
                        else
                           put (wfgap+wf-(i8(temphirert)*hiredy)+
                                ((i12(tempnewtrr)+i13(tempexptrr))*trnsdy)) into wfgap
                        end if
                        put 0 into (newtrrstock)i10
                        put 0 into (exptrrstock)i11
                        put 0 into (hirestock)i9
                        loop = 0
                        cont = 1
                  else
                        cont = 0
                  end if
            else
                  cont = 0
            end if
      else cont = 0
      end if


IF CONT = 1

period = 1

    put "PROCESSING ........." into ad31
    select range ad31
    text size 14
    text style "I"
    text color Blue()
    unselect

while period <= inputperiod and ptktst < 0.99 and

(cummd<=2500) and (time <= 500) (check point)

    put (cumtkt/(cmtsmd+0.001)) into actspd

    put cmtkdv/rjbsz into pjbawk

    put wfexp+wfnew into totwf
    put totwf*admpps into totdmp

    put wfnew*trpnhr into dmptrn
    put totdmp-dmptrn into dmpatr
    put min((afmpqa*totdmp),0.9*dmpatr) into dmpqa
    put dmpqa into ab22
    put dmpatr-dmpqa into dmpswp
    put perdmprw*dmpswp into dmprw
    put dmprw into ab23

put dmpswp-dmprw into dmpdvt
put dmpdvt into ab20

put (cmtkdv/pjbsz)*100 into pjbpwk
put pjbpwk into ad5

if (pjbpwk/100) > 1
    put 0 into mpwdev
elseif (pjbpwk/100) > 0.9
    put (-5*(pjbpwk/100)+5) into mpwdev
elseif (pjbpwk/100) > 0.8
    put (-2.5*(pjbpwk/100)+2.75) into mpwdev
elseif (pjbpwk/100) > 0.7
    put (-1.5*(pjbpwk/100)+1.95) into mpwdev
elseif (pjbpwk/100) > 0.6
    put (-0.75*(pjbpwk/100)+1.425) into mpwdev
elseif (pjbpwk/100) > 0.5
    put (-0.25*(pjbpwk/100)+1.125) into mpwdev
else put 1 into mpwdev
end if
```

```
put cmtkdv/(cummd+cmtsmd) into prdprd
put pjbsz-cumtkt into tskprm
put tsktprm/prtprd into mdpnts
put dtcerr*prwmpe into mdpnrw
put max(0.0001,jbszmd-cummd) into mdrm
put max(0,mdrm-mdpnrw-mdpnts) into mdprnt

if (1-mdprnt/(jbszmd-tszzmd)) > 1
   put 0 into mpwrex
elseif (1-mdprnt/(jbszmd-tszzmd)) > 0.9
   put (-5*(1-mdprnt/(jbszmd-tszzmd))+5) into mpwrex
elseif (1-mdprnt/(jbszmd-tszzmd)) > 0.8
   put (-2.5*(1-mdprnt/(jbszmd-tszzmd))+2.75) into mpwrex
elseif (1-mdprnt/(jbszmd-tszzmd)) > 0.7
   put (-1.5*(1-mdprnt/(jbszmd-tszzmd))+1.95) into mpwrex
elseif (1-mdprnt/(jbszmd-tszzmd)) > 0.6
   put (-0.75*(1-mdprnt/(jbszmd-tszzmd))+1.425) into mpwrex
elseif (1-mdprnt/(jbszmd-tszzmd)) > 0.5
   put (-0.25*(1-mdprnt/(jbszmd-tszzmd))+1.125) into mpwrex
else put 1 into mpwrex
end if

put mpwdev*mpwrex into wtpjdp
put (pjbsz-cmtkdv) into tskprm
put tskprm/(mdprnt+0.1) into pjdprd
put pjdprd*wtpjdp+prdprd*(1-wtpjdp) into asspprd
put (tskprm/asspprd) into mdpnnt
put (mdpnnt+mdpnts+mdpnrw) into tmdpsn
put (tmdpsn-mdrm)/mdrm into schpr

put tskwk1+tskwk2 into tskwk

put max(ptdter/(tskwk+0.0001),0) into anerpt

put wfexp/totwf into frwfex
put frwfex*npwpex+(1-frwfex)*npwpnm into anppprd
put wfexp*admpps into ftexwt
put ftexwt*mnhpxs into celnwh
put celnwh+wfexp into celtwt


if (tskprm/pjbsz) > 0.2
   put 0 into trefts
elseif (tskprm/pjbsz) > 0.16
   put (-1.25*(tskprm/pjbsz)+0.25) into trefts
elseif (tskprm/pjbsz) > 0.12
   put (-2.5*(tskprm/pjbsz)+0.45) into trefts
elseif (tskprm/pjbsz) > 0.08
   put (-3.25*(tskprm/pjbsz)+0.54) into trefts
elseif (tskprm/pjbsz) > 0.04
   put (-5.5*(tskprm/pjbsz)+0.72) into trefts
elseif (tskprm/pjbsz) > 0
   put (-12.5*(tskprm/pjbsz)+1) into trefts
else put 1 into trefts
end if
put dmpdvt*(1-trefts) into dmpskiv

put anerpt*1000/dslptk into errdsy


if pjbawk > 1
   put 12.5 into nerpk
elseif pjbawk > 0.8
   put (-5.5*pjbawk+18) into nerpk
elseif pjbawk > 0.6
   put (-11.5*pjbawk+22.8) into nerpk
elseif pjbawk > 0.4
   put (-28.5*pjbawk+33) into nerpk
elseif pjbawk > 0.2
   put (-11.5*pjbawk+26.2) into nerpk
elseif pjbawk > 0
   put (-5.5*pjbawk+25) into nerpk
else put 25 into nerpk
end if

if schpr > 1
   put 1.5 into mergsp
elseif schpr > 0.8
   put (0.7*schpr+0.8) into mergsp
elseif schpr > 0.6
   put (0.6*schpr+0.88) into mergsp
elseif schpr > 0.4
   put (0.5*schpr+0.94) into mergsp
elseif schpr > 0.2
   put (0.45*schpr+0.96) into mergsp
elseif schpr > 0
   put (0.25*schpr+1) into mergsp
elseif schpr > -0.2
   put (0.3*schpr+1) into mergsp
elseif schpr > -0.4
   put (0.2*schpr+0.98) into mergsp
else put 0.9 into mergsp
end if

if frwfex > 1
   put 1 into mergwm
elseif frwfex > 0
   put (-1*frwfex+2) into mergwm
else put 2 into mergwm
end if

put nerpk*dslptk/1000 into nerplk
put nerptk*mergsp*mergwm into nerplk
```

```
put totwf*admpps into fteqwf
put tmdpsn-mdrm into pmissc

if time >= (brkdtm+rlxtmc)
    put 1 into wtovwk
else put 0 into wtovwk
end if

if (exhlev/mxexht) > 1
    put 0 into modtex
elseif (exhlev/mxexht) > 0
    put (-1*exhlev/mxexht+1) into modtex
else put 1 into modtex
end if

put max(schedt-time,0) into timerm
if timerm > 50
    put 50 into novwit
elseif timerm > 0
    put timerm into novwit
else put 0 into novwit
end if

put novwit*modtex into ovwith
put ovwith*fteqwf*maxmhr*wtovwk into maxshr

if (tmdpsn/mdrm) > 0.8
    put 1 into texabs
elseif (tmdpsn/mdrm) > 0.7
    put (0.5*tmdpsn/mdrm+0.6) into texabs
elseif (tmdpsn/mdrm) > 0.6
    put (0.5*tmdpsn/mdrm+0.6) into texabs
elseif (tmdpsn/mdrm) > 0.5
    put (1*tmdpsn/mdrm+0.3) into texabs
elseif (tmdpsn/mdrm) > 0.4
    put (1*tmdpsn/mdrm+0.3) into texabs
elseif (tmdpsn/mdrm) > 0.3
    put (1.5*tmdpsn/mdrm+0.1) into texabs
elseif (tmdpsn/mdrm) > 0.2
    put (1.5*tmdpsn/mdrm+0.1) into texabs
elseif (tmdpsn/mdrm) > 0.1
    put (2*tmdpsn/mdrm) into texabs
elseif (tmdpsn/mdrm) > 0
    put (2*tmdpsn/mdrm) into texabs
else put 0 into texabs
end if

put max(0,(texabs*mdrm-tmdpsn)) into exsabs
if pmdshr >= 0
    put min(maxshr,pmdshr)*ctrlsw into mdhdl
else put (-exsabs)*ctrlsw into mdhdl
end if

if pmdshr >=0
    put (mdhdl/(fteqwf*(ovwith+0.0001))) into pbwkrs
else put (mdhdl/(tmdpsn-mdhdl+0.0001)) into pbwkrs
end if

put (1+pbwkrs)*mfmdpj into wkrts
if wkrts >= afmdpj
    put 1 into ewkrts
else put 0.75 into ewkrts
end if


if totwf > 30
    put 0.54 into commoh
elseif totwf > 25
    put (0.033*totwf-0.45) into commoh
elseif totwf > 20
    put (0.027*totwf-0.3) into commoh
elseif totwf > 15
    put (0.021*totwf-0.18) into commoh
elseif totwf > 10
    put (0.015*totwf-0.09) into commoh
elseif totwf > 5
    put (0.009*totwf-0.03) into commoh
elseif totwf > 0
    put (0.003*totwf) into commoh
else put 0 into commoh
end if
put afmdpj*(1-commoh) into mpdmcl

if pjbawk > 0.8
    put 1.25 into mpptpd
elseif pjbawk > 0.7
    put (0.3*pjbawk+1.01) into mpptpd
elseif pjbawk > 0.6
    put (2*pjbawk-0.18) into mpptpd
elseif pjbawk > 0.5
    put (0.5*pjbawk+0.9) into mpptpd
elseif pjbawk > 0.4
    put (0.6*pjbawk+0.85) into mpptpd
elseif pjbawk > 0.3
    put (0.4*pjbawk+0.93) into mpptpd
elseif pjbawk > 0.2
    put (0.2*pjbawk+0.99) into mpptpd
elseif pjbawk > 0.1
    put (0.2*pjbawk+0.99) into mpptpd
elseif pjbawk > 0
    put (0.1*pjbawk+1) into mpptpd
else put 1 into mpptpd
end if

put anpprd*mpptpd into potpprd
```

```
put disc1+disc2 into tkdscv
put tkdscv/asspr4 into pszdct
put cumtkt/pjbsz into ptktst
put (ptktst*100) into af5

put pszdct/(mdprnt+0.0001) into rszdct

if pjbawk > 1
   put 0.224 into nrwmpe
elseif pjbawk > 0.8
   put (-0.25*pjbawk+0.474) into nrwmpe
elseif pjbawk > 0.6
   put (-0.27*pjbawk+0.49) into nrwmpe
elseif pjbawk > 0.4
   put (-0.225*pjbawk+0.463) into nrwmpe
elseif pjbawk > 0.2
   put (-0.21*pjbawk+0.457) into nrwmpe
elseif pjbawk > 0
   put (-0.205*pjbawk+0.456) into nrwmpe
else put 0.456 into nrwmpe
end if

put nrwmpe/mpdmcl into rwmppe
put potprd*mpdmcl into sdvprd

put pjbsz/tszzmd into pltspd
if cumtkt > 0
   put actspd into select
else put pltspd into select
end if

put pmdshr-mdhdl into shrrpt
put underr/(cumtqa+0.1) into underrdsy
put dsystock into smterrdsy

if ovwdth = 0
   put time+dt into sw
else put 0 into sw
end if


put (mdrm/(timerm+0.001))/admpps into wfindic

put max(0,wfgap/hiredy) into hirert
put hirert into (temphirert)18

put (hirestock)19+hirert*dt into (hirestock)19
if (hirestock)19 >= wfgap
   put 0 into hirert
end if

put min(max(0,-wfgap/trnsdy),wfnew/dt) into newtrr
put newtrr into 112(tempnewtrr)
put (newtrrstock)110+newtrr*dt into (newtrrstock)110
if (newtrrstock)110 >= abs(wfgap)
   put 0 into newtrr
end if

put min(wfexp/dt,max(0,-wfgap/trnsdy)-newtrr) into exptrr
put exptrr into (tempexptrr)113
put (exptrrstock)111+dt*exptrr into (exptrrstock)111
if (exptrrstock)111 >= abs(wfgap)
   put 0 into exptrr
end if

put wf+totwf into wfs
put mdrm/(wfs*admpps) into timepr

if timerm > 30
   put 10 into nwrady
elseif timerm > 25
   put (0.1*timerm+7) into nwrady
elseif timerm > 0
   put (0.3*timerm+2) into nwrady
else put 2 into nwrady
end if
put nwrady*ewkrts into wkrady
put (tstovh*dsiptk/1000+tmpnpe*underrdsy)/mpdmcl into tmpnpt
put mdrm+shrrpt into mdrptm
put time+timepr into indcdt

{RATES}

put (underrdsy-dsystock)/tsaeds into dsyrate
put (wkrts-afmdpj)/wkrady into wradjr
put (max(brkdtm,sw)-brkdtm)/dt into breakdown
put min((dmpsdv*sdvprd),tskprm/dt) into sdvrt1
put sdvrt1 into sdvrt2
put dmpdvt*frefts into dmptst
put dmptst into ab21

put min(cumtqa/dt,dmptst/tmpnpt) into tsrate
put tskwk2/aqadly into qart2

if pjbpwk > 100
   put 100 into putdpd
elseif pjbpwk > 80
   put (4.5*pjbpwk-350) into putdpd
elseif pjbpwk > 60
   put (0.25*pjbpwk-10) into putdpd
elseif pjbpwk > 40
   put (0.125*pjbpwk-2.5) into putdpd
elseif pjbpwk > 20
   put (0.105*pjbpwk-1.7) into putdpd
```

```
elseif pjbpwk > 0
    put (0.02*pjbpwk) into putdpd
else put 0 into putdpd
end if
put undjtk*putdpd/100 into rtdstk

put disc1/dinct into rtinct1
put disc2/dinct into rtinct2

if afmpqa > 0.7
    put 0.905 into qadjust
elseif afmpqa > 0.6
    put (0.05*afmpqa+0.87) into qadjust
elseif afmpqa > 0.5
    put (0.5*afmpqa+0.6) into qadjust
elseif afmpqa > 0.4
    put (0.55*afmpqa+0.575) into qadjust
elseif afmpqa > 0.3
    put (1.05*afmpqa+0.375) into qadjust
elseif afmpqa > 0.2
    put (1.35*afmpqa+0.285) into qadjust
elseif afmpqa > 0.1
    put (2.3*afmpqa+0.095) into qadjust
elseif afmpqa > 0
    put (3.2*afmpqa+0.005) into qadjust
else put 0.005 into qadjust
end if
put ptdter*qadjust into errdrt

put min(dmprw/rwmppe,dtcerr/dt) into rwrate

if (1-afmdpj)/(1-nfmdpj) > 0.9
    put 0 into riexhl
elseif (1-afmdpj)/(1-nfmdpj) > 0.8
    put (-2*((1-afmdpj)/(1-nfmdpj))+1.8) into riexhl
elseif (1-afmdpj)/(1-nfmdpj) > 0.1
    put (-1*((1-afmdpj)/(1-nfmdpj))+1) into riexhl
elseif (1-afmdpj)/(1-nfmdpj) > 0
    put (-2.5*((1-afmdpj)/(1-nfmdpj))+1.15) into riexhl
elseif (1-afmdpj)/(1-nfmdpj) > -0.1
    put (-1.5*((1-afmdpj)/(1-nfmdpj))+1.15) into riexhl
elseif (1-afmdpj)/(1-nfmdpj) > -0.5
    put (-3*((1-afmdpj)/(1-nfmdpj))+1) into riexhl
else put 2.5 into riexhl
end if

if riexhl > 0
    put 0 into rdexhl
else put exhlev/exhddy into rdexhl
end if

if (rszdct/(msztwo+0.001)) > 1.8
    put 1 into fadhwo
elseif (rszdct/(msztwo+0.001)) > 1.6
    put (0.125*(rszdct/(msztwo+0.001))+0.775) into fadhwo
elseif (rszdct/(msztwo+0.001)) > 1.4
    put (0.375*(rszdct/(msztwo+0.001))+0.375) into fadhwo
elseif (rszdct/(msztwo+0.001)) > 1.2
    put (1*(rszdct/(msztwo+0.001))-0.5) into fadhwo
elseif (rszdct/(msztwo+0.001)) > 1
    put (3.5*(rszdct/(msztwo+0.001))-3.5) into fadhwo
else put 0 into fadhwo
end if

if timerm > 20
    put 3 into dajbmd
elseif timerm > 0
    put (0.125*timerm+0.5) into dajbmd
else put 0.5 into dajbmd
end if

put (rtinct2/asspd)*fadhwo into irdvdt
put (rtinct2/prtprd)*fadhwo into irtsdt1
put (mdrptn+cummd-jbszmd)/dajbmd into artjbm
put (select-prtprd)/tstspd into dummyrate
put (rwmppe-prwmpe)/tarmpe into change1
put sdvrt1*errptk into errgrt
put 0.1*ptdter into errsrt

if (exhlev/mxexht) >= 0.1
    put 1 into deexhaust
else put (-rlxtmc/dt) into deexhaust
end if

if ovwdth = 0
    put (rlxtmc/dt) into discharge
else put 0 into discharge
end if

if timerm > 5
    put 10 into schadt
elseif timerm > 0
    put (1.9*timerm+0.5) into schadt
else put 0.5 into schadt
end if
put (indcdt-schcdt)/schadt into change3

put tskwk1/aqadly into qart1
put irtsdt1 into irtsdt2

if frefts >= 0.9
    put (artjbm/dt) into tszrate
else put 0 into tszrate
end if
```

```
put sdvrt1*smterrdsy into rgnrt
put min(tsrate*underrdsy,underr/dt) into correct
put wfnew/asimdy into asimrt
put wfexp/avempt into quitrt


(STOCKS)


put dsystock+dt*dsyrate into dsystock
put afmdpj+dt*wradjr into afmdpj
put brkdtm+dt*breakdown into brkdtm
put cmtkdv+dt*sdvrt2 into cmtkdv
put cmtkdv into ae23
put cmtsmd+dt*dmptst into cmtsmd
put cummd+dt*totdmp into cummd
put cummd into ac5

put cumtkt+dt*tsrate into cumtkt
put cumtkt into ae24
put cumtqa+dt*(qart2-tsrate) into cumtqa
put disc1+dt*(rtdstk-rtinct1) into disc1
put disc2+dt*(rtinct1-rtinct2) into disc2
put dtcerr+dt*(errdrt-rwrate) into dtcerr
put exhlev+dt*(riexhl-rdexhl) into exhlev
put jbszmd+dt*(irdvdt+irtsdt1+artjbm) into jbszmd
put jbszmd into ae22
put pjbsz+dt*rtinct2 into pjbsz
put pjbsz into ae21
put prtprd+dt*dummyrate into prtprd
put prwmpe+dt*change1 into prwmpe
put ptdter+dt*(errgrt-errsrt-errdrt) into ptdter
put rlxtmc+dt*(deexhaust-discharge) into rlxtmc
put schcdt+dt*change3 into schcdt
put schcdt into aa5

put tskwk1+dt*(sdvrt1-qart1) into tskwk1
put tskwk2+dt*(qart1-qart2) into tskwk2
put tszzmd+dt*(irtsdt2+tszrate) into tszzmd
put underr+dt*(errsrt+rgnrt-correct) into underr
put undjtk+dt*(-rtdstk) into undjtk
put wfexp+dt*(asimrt-quitrt-exptrr) into wfexp
put wfexp into ae20
put wfnew+dt*(hirert-asimrt-newtrr) into wfnew
put wfnew into ae19
put time+dt into time
put time into ab31
select range ab31..ab31
precision(0)
unselect
period = period+1


END WHILE

if (cummd > 2500) or (time > 500)
    if (time > 500) and (cummd > 2500)
        run script "c:\wing2\bothout.scz"
    else if (cummd > 2500)
        run script "c:\wing2\budgout.scz"
    else if (time > 500)
        run script "c:\wing2\timeout.scz"
    end if
    end if
    end if
else if (cummd <= 2500) and (time <= 500) and (ptktst>=.99)
        run script "c:\wing2\congr.scz"
end if
end if

put time into makecell(105,r)
put wfnew into makecell(107,r)
put wfexp into makecell(108,r)
put dmptrn into makecell(109,r)
put totdmp into makecell(110,r)
put cummd into makecell(111,r)
put dmpqa into makecell(112,r)
put dmprw into makecell(113,r)
put dmpdvt into makecell(114,r)
put dmpsdv into makecell(115,r)
put frefts into makecell(116,r)
put sdvprd into makecell(117,r)
put exhlev into makecell(118,r)
put errgrt into makecell(119,r)
put errdrt into makecell(120,r)
put dtcerr into makecell(121,r)
put underr into makecell(122,r)
put dmptst into makecell(123,r)
put cmtsmd into makecell(124,r)
put ptktst into makecell(125,r)
put cumtkt into makecell(126,r)
put cmtkdv into makecell(127,r)
put tmdpsn into makecell(128,r)
put tstprm into makecell(129,r)
put prtprd into makecell(130,r)
put pjbpwk into makecell(131,r)
put pjbsz into makecell(132,r)
put tskprm into makecell(133,r)
put tszzmd into makecell(134,r)
put jbszmd into makecell(135,r)
put schcdt into makecell(136,r)
put timerm into makecell(137,r)
put r+1 into rtemp
```

```
select object 3
clear
select more object 4
clear
select more object 5
clear
select more object 6
clear
unselect

run script "c:\wingz\chart.scz"
put "                              " into edit
END IF
```

```
{ CONSTANTS AND INITIAL VALUES FOR STOCKS }

select graphics
clear
unselect
select value cells
clear
unselect

put 0 into da1
put 0 into dc1
put 3 into dl1
put 1 into dm1
put 0 into ea1

put 15 into y99
put 50 into y98
put 0 into y97

put 2 into rtemp
put 10 into inputperiod
put 0 into i12{tempnewtrr}
put 0 into i13{tempexptrr}
put 0 into i8{temphirert}
put 0 into wfgap
put 0 into dsystock
put 0 into time
put time into ab31
select range ab31..ab31
precision(0)
unselect

put 1 into dt
put 0.6 into nfmdpj
put nfmdpj into afmdpj
put -1 into brkdtm

put 0 into cmtkdv
put cmtkdv into ae23
put cmtkdv into dw1

put 0 into cmtsmd
put 0 into dt1

put 0.0001 into cummd
put cummd into ac5
select range ac5
align left
unselect
put cummd into dg1

put 0 into cumtkt
put cumtkt into ae24
put cumtkt into dv1

put 0 into cumtqa
put 0 into disc1
put 0 into disc2
put 0 into dtcerr
put 0 into dq1

put 0 into exh1ev
put 0 into dn1

put 0.8 into devprt
put 1 into mdswch
put 0 into undest
put 25000 into rjbdsi
put (rjbdsi*(1-undest)) into pjbdsi
put 0 into undesm
put 0 into totmd1
put mdswch*(((2.4*exp(1.05*ln(pjbdsi/1000)))*19)*(1-undesm))+
    (1-mdswch)*totmd1 into totmd
put(devprt*totmd) into devmd
put ((1-devprt)*totmd) into tstmd

put (devmd+tstmd) into jbszmd
put jbszmd into ae22
put jbszmd into ee1

put 60 into dsiptk

put (pjbdsi/dsiptk) into pjbsz
put pjbsz into ae21
put pjbsz into eb1

put pjbsz-cmtkdv into ec1

put pjbsz-cumtkt into dy1


put tstmd into tszzmd
put tszzmd into ed1

put (pjbsz/tszzmd) into pltspd
put pltspd into select
put select into prtprd
put 0.5 into prwmpe
put 0 into ptdter
put 0 into rlxtmc
put 1 into scswch
put 1 into schcom
put 0 into tdev1
put (scswch*((19*2.5*exp(0.38*ln(totmd/19)))*schcom)+(1-scswch)*tdev1) into tdev
```

```
put tdev into schcdt
put schcdt into aa5
select range aa5
align center
unselect
put schcdt into efl
put schcdt into egl

put 0 into tskwkl
put 0 into tskwk2
put 0 into underr
put 0 into drl

put (rjbdsi/dslptk) into rjbsz

put (cmtkdv/rjbsz) into pjbawk
put pjbawk into ad5
select range ad5
align center
unselect

put (rjbsz-pjbsz) into undjtk
put 0.5 into inudst
put 1 into admpps
put ((totmd/tdev)/admpps) into teamsz
put (teamsz*inudst) into wfstrt

put wfstrt into wfexp
put wfexp into ae20
put wfexp into ddl

put 0 into wfnew
put wfnew into ael9

put wfnew+wfexp into totwf
put totwf*admpps into totdmp
put totdmp into dfl

put 5 into aqadly
put 80 into asimdy
put 673 into avempt
put 1 into ctrlsw
put 5 into dlinct
put 20 into exhddy
put 10 into hiredy
put 1 into maxmhr
put 3 into mnhpxs
put 0.01 into msztwo
put 50 into mxexht
put 1 into npwpex
put 0.5 into npwpne
put 10 into tarmpe
put 0.25 into tmpnpe
put 10 into trnsdy
put 0.2 into trpnhr
put 40 into tsaeds
put 1 into tstovh
put 50 into tstspd
put 15 into desrwd

put (cumtkt/pjbsz) into ptktst
put ptktst into af5
put ptktst into dul

run script "c:\wingz\chart.scz"
```