AN INTERACTIVE SIMULATION GAME FOR SOFTWARE PROJECT MANAGEMENT (SOFTSIM)

by
Yaman Barlas
Ibrahim Bayraktutar

Department of Systems Analysis
Miami University
Oxford, OHIO 45056

ABSTRACT

Various uses of system dynamics models in understanding and managing software projects have been discussed in a series of articles by Tarek Abdel-Hamid and Stuart Madnick. Our work extends Abdel-Hamid and Madnick's work by constructing a simulation game that can be used by managers as a participatory learning laboratory. The game is implemented on IBM-PC environment, using the graphics-based spreadsheet software WingZ. Initial tests of the simulation game have demonstrated it to be robust and realistic. The game is now in the process of being tested extensively by players with different backgrounds: undergraduate and graduate students of systems analysis, faculty members, and software/MIS professionals. In addition to describing the game, our talk will contain lessons learned from these extended tests and experiments. The presentation will also involve an active demonstration.

INTRODUCTION

Software industry is one of the most rapidly growing businesses in our age. Yet, as Abdel-Hamid and Madnick (1991) point out, this growth has not been a well-balanced one. On the technical side, programming languages, techniques and methodologies have exhibited an unprecedented growth. But on the other hand, the art and science of managing software projects has not enjoyed such a growth. Abdel-Hamid and Madnick (1991) state that management of software systems has been plagued by cost overruns, late deliveries and poor reliability. They attribute this situation to a lack of fundamental understanding of the software development process, caused by lack of research on the managerial aspects of software development (Abdel-Hamid and Madnick (1989, 1991)). The comprehensive system dynamics model of software management developed by Abdel-Hamid and Madnick seeks to fulfill this need. The purpose of their simulation model is to "enhance our understanding of" and "provide insight into" the management of software development. Our work extends Abdel-Hamid and Madnick's work by constructing an interactive simulation game based on their comprehensive system dynamics model. There is a growing trend in system dynamics community to transform system dynamics models into dynamic interactive games, in order to create "reflective learning environments" (Kim (1989), Meadows (1989) and Andersen et.al. (1990)). It is believed that such interactive simulation games promote active learning, convey principles of systems, and provide context for systems research and evaluation (Kim (1989) and Graham et.al (1989)). Our purpose is to design an interactive game in which players make some key software management decisions that determine the course of the simulation. Players' objective is to complete a given software project within a given time, subject to a given budget limitation. In seeking this goal, players get a chance to test various strategies in this "laboratory," without risking actual budget overruns or late deliveries. Players are expected to gain insights into the effectiveness of various management strategies, by adopting some sort of "systematic trial and error" approach. The ultimate objective of this project is to use the simulation game as an interactive learning laboratory to train software management students and professionals.

DESCRIPTION OF THE GAME

The software management simulation game (SOFTSIM) consists of three parts: 1- the system dynamics model which drives the simulation in the background, 2- the decision controls, which is the summary information screen where the player either makes decisions or chooses to see more detailed information, 3- the information system that organizes and presents the data the player wishes to analyze.

The system dynamics model behind the interactive game has essentially the same structure as Abdel-Hamid and Madnick's model. Thus, like their model, our model too focuses only on the development phases of software production. (i.e. the initial software planning/definition and the very last software maintenance phases are not included). The model consists of seven major sectors: Human resource management, manpower allocation, software development, quality assurance and rework, system testing, and controlling/planning. These sectors are naturally interconnected, as illustrated in Figure 1. For more information on the individual variables and equations involved in these different sectors, see Abdel-Hamid and Madnick (1991). Our model differs from theirs in two aspects: first, we simplified their model to some extent in order to increase the speed of the gaming version. We made sure that these simplifications did not change the behavior of the model in a significant way, especially with respect to the purpose of the game. Second, we had to make several changes in those equations that are directly related to interactive player decisions. These variables will be described in the next paragraph. Readers who are interested in these technical differences between Abdel-Hamid and Madnick's model and our version are referred to Bayraktutar (1992) and Abdel-Hamid and Madnick (1991). Let us finally note that we have extensively tested our version under numerous (mostly "extreme") conditions and found its structure to be quite robust. We have also verified the accuracy of the interactive gaming version, by entering the decisions internally made by the closed-loop system dynamics model, as synthetic "interactive player inputs." In all cases, the behavior patterns obtained by the decisions-in-the-loops version and the interactive gaming version were identical. (See Bayraktutar (1992)).

The "decision controls" screen, analogous to an aircraft's cockpit, displays the most important information, and allows the player to make decisions or request more detailed information (figure 2). Four crucial summary indicators (Scheduled Completion Date, Cumulative Man-days, % Development Tasks Completed and % Testing Tasks Completed) are displayed in graphical form for maximum clarity. Information on ten other important variables is also included in the same screen. (Issues involved in interactive gaming screen and other aspects of game design are discussed in Andersen et.al.(1990) and Meadows (1989)). In the controls screen, the player examines the key summary indicators, and then either makes a decision ("make decision" button) or chooses to analyze more data ("more information" button). Once she feels ready, she is asked to make three decisions: 1- % man power to be allocated for quality assurance, 2- % man-power to be allocated for rework (the remaining man-power is used in development and testing), and 3- staff addition or removal. (See the decision input windows in figure 3). Once all three decisions are made, the simulation model runs for a short period of time (this "decision period" is either 10 or 20 days, depending on the size of the project). At the end of the decision period, the player is presented with the current summary information, resulting from her most recent decisions. And this process repeats itself until the game ends.

If, in the controls screen, the player wishes to analyze more information before he makes decisions, he must access the "information system" by clicking on "more information" button. She is then presented with a long list of additional model variables to choose from (the scroll-bar list shown in figure 4). The player then chooses variables (one at a time) from this list, and examines their history and current values. (Figure 5 illustrates what kind of information the player, upon choosing a variable, is presented with). After having analyzed information on various variables, once the player feels ready to make decisions, she clicks on "back to main" which returns her back to the main controls screen where decisions are made.

The game is started by running the WingZ program called SOFTSIM. In the opening screen, the game offers various options for the player to choose from: Small project versus large project; project size exactly known versus project size underestimated; and "easy" version (hiring delay=decision period) versus "difficult" version (hiring delay ≠ decision period). Any one of these options can be selected by pulling down the "options" menu. The most difficult version of the game is obtained when the project size is underestimated and the hiring delay ≠ decision period.

The objective of playing SOFTSIM is to complete the given software project within the given time and budget constraints. If a player can accomplish this, he gets a congratulation message at the end. But there are also three undesirable ways in which the game may end: the player may exceed the given time limit, she may exceed the given budget limit, or worse, he may exceed both the time and budget limits. In any of these three cases, the player is shown at the end a message indicating the problem.

We have tested the validity of SOFTSIM, by entering a wide range of synthetic player decisions (most of which extreme and unlikely). The game proved to be quite robust and reliable, even under extreme decision inputs. We then extended these tests, by exposing the game to external criticisms by independent players. None of the ten players who tested the game raised any substantial criticism of the structure of SOFTSIM. In each of these ten experiments, SOFTSIM yielded behavior patterns that were plausible, realistic, and consistent with the decisions made by players. The only major criticism of the game (which we agree with) is its speed. Currently, a typical game session with the "small" project option lasts about 1.5 hours. For the game to be of practical use, this speed must be increased, possibly doubled.

EXPERIMENTAL RESULTS

We now present information on performances of eight different players (two systems analysis faculty members, four graduate students and two undergraduate students). Table 1 summarizes performances of these eight players at the end of the game, as well as the performance of the decision-in-the-loops version of the simulation model. Observe in the first row of Table 1, that as many as six out of eight players finished the project earlier than the model did. But, none of these eight players was able to complete the project with less budget than the model did (the fourth row in Table 1). As a matter of fact, several players came pretty close to the bankruptcy limit (2500 man-days). There is a natural trade-off between completing early and staying within the budget. It seems that all of the eight players put much more emphasis on completing early, than they did on controlling expenditures, although both factors were equally emphasized during game briefing. (Could this consistent bias of over-focusing on time constraints be a cultural one? Could it be an artifact of the contemporary, fast-paced way of life, where most decisions are dictated by sort-term time constraints? We leave such questions to the experts).

The last four rows of Table 1 display actual decisions made by the players (and the model) at the end of the game. As expected, Manpower allocated to Quality Assurance, Manpower allocated to Rework and New Workforce are all zero, since, toward the end of the project, all development (hence quality assurance and most of rework) activities are complete, and the entire workforce is allocated to testing. (An exception to this rule are players 4, 5 and 8 who have a small fraction of their manpower allocated to rework at the end). Note also that the Daily Manpower for Development/testing (DMPDVT in row 7) actually corresponds to testing exclusively, since there is no development activities at the end of the project. The very last row represents the ending (experienced) workforce level (new workforce is naturally zero at the end), which displays substantial variation, ranging from 1.22 (player 3) to 13.45 (player 4), and the model's being 4.53. Once again, observe that the entire ending workforce is devoted to Daily Manpower for Testing (row 7), with the exception of players 4, 5 and 8, for reasons explained above.

It is also informative to examine the dynamic behavior patterns generated by the players through the game. Due to space limitations, we are only able to show one illustration.

Figure 6 depicts the behavior patterns generated by player 1. Figure 6a displays the dynamics of five major variables. (Perceived Job Size, PJBSZ never changes, since in the simpler version of the game, it is assumed that the exact project size is known). Cumulative Man Days (CUMMD) increases linearly, and the project is completed when it catches up with Job Size in Man Days (JBSZMD). Notice that there is a late increase in JBSZMD, that represents the fact that, toward the end of the project, realizing that we have been overestimating our productivity, we correct our estimate of JBSZMD upward. Note that, when Cumulative Tasks Developed (CMTKDV) catches up with PJBSZ, the project is not yet complete; this is the time when (almost) all manpower gets allocated to testing. This can be more clearly seen in Figure 6b: The point in time at which CMTKDV=PJBSZ, we see a significant drop in Daily Manpower for Quality Assurance (DMPQA) and Daily Manpower for Rework (DMPRW), and a significant jump in Daily Manpower for Testing (DMDVT). Finally, WFNEW in Figure 6b represents the player's hiring policy. The player hires more than enough staff very early in the project, and then gradually dismisses them as the project gets completed. This initial jump in the workforce level also causes the early estimates of the Scheduled Completion Date (SCHCDT) to be lower than later realized. (Compare curve 4 of Figure 6b and curve 1 of Figure 6a).

CONCLUSIONS

We have developed and tested an interactive software management game (SOFTSIM) based on Abdel-Hamid and Madnick's model. Experiments demonstrate that SOFTSIM responds to player decisions in a robust, meaningful and realistic way. Reactions of players have been in general very positive. One weakness that has been observed is the speed of the game. For the game to be used as a practical learning laboratory, its speed must be substantially increased.

The game also promises to be a useful platform to test different theories of dynamic decision making in the context of software management. For instance, Sterman (1989) carries out such a decision-theoretic research using an interactive macroeconomic simulator. SOFTSIM promises a platform to extend/replicate the research in the area of software management. Finally, using SOFTSIM, one can use direct experimentation to test the validity of decision equations used in Abdel-Hamid and Madnick's model. (See Sterman (1987)). Our initial tests show that behavior patterns generated by players and those generated by the non-interactive version of the model display remarkable similarities.

REFERENCES

Bayraktutar, Ibrahim. 1992. *A Simulation Game for Software Project Management*, Master's Report, Department of Systems Analysis, Miami University, Oxford, OHIO.

Andersen, D.F., Chung, I.J, Richardson, G.P and Stewart, T.R. 1990. Issues In Designing Interactive Games Based on System Dynamics Models, *Proceedings of International System Dynamics Conference*: 31-45

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

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

Sterman, J.D.1987. Testing Behavioral Simulation Models by Direct Experiment. *Management Science* 33: 1572-1592

Sterman, J.D.1989. Misperceptions of Feedback in Dynamic Decision Making. *Organizational Behavior and Human Decision Processes* 43:301-335.

Abdel-Hamid, T.K. and S. E. Madnick. 1991. *Software Project Dynamics, An Integrated Approach*. New Jersey: Prentice-Hall.

Abdel-Hamid, T.K. and S.E. Madnick. 1989. Lessons Learned from the Dynamics of Software Development. *Communications of the ACM*. 32: 1426-1438

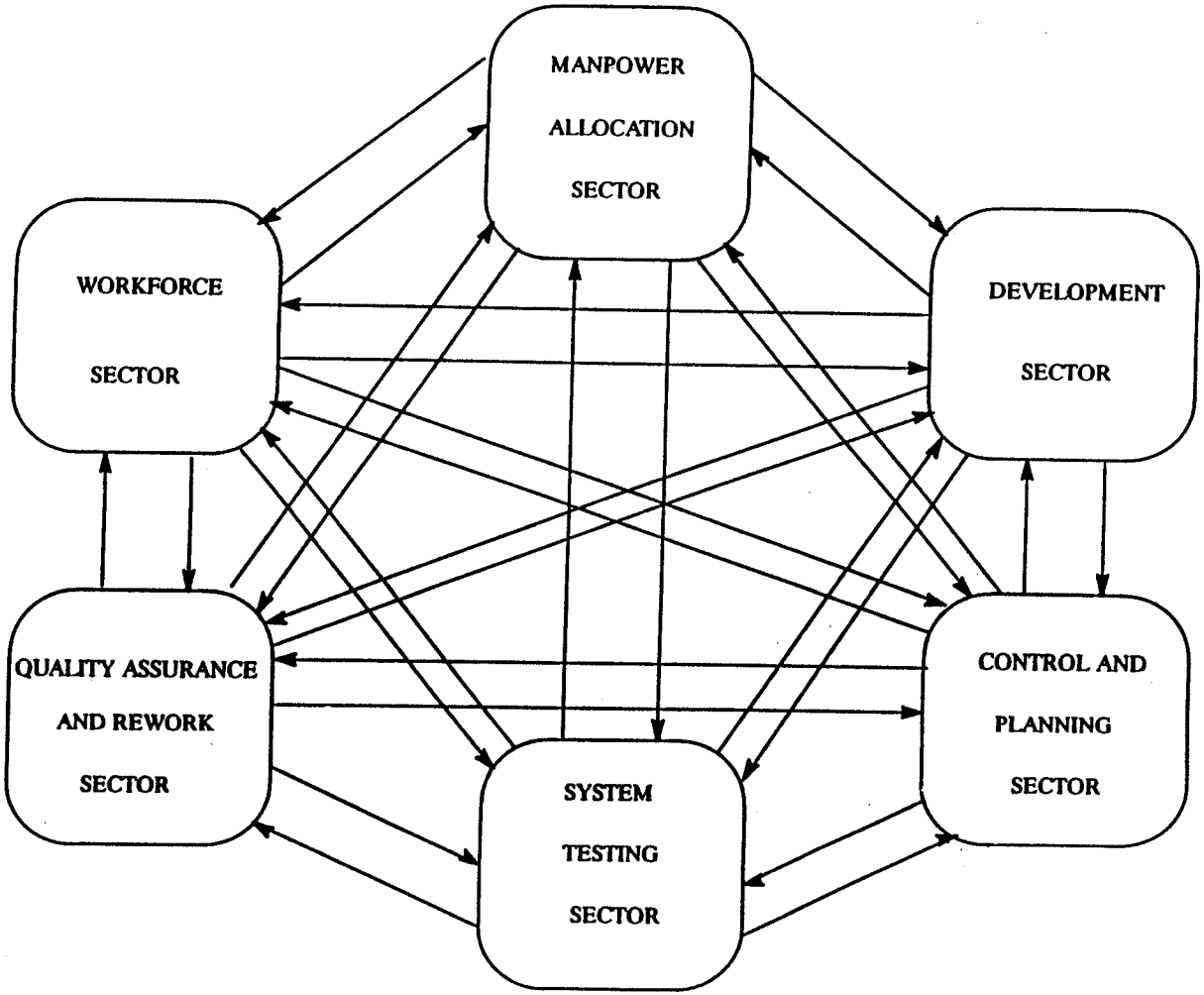

Figure 1. An Overview of the Simulation Model

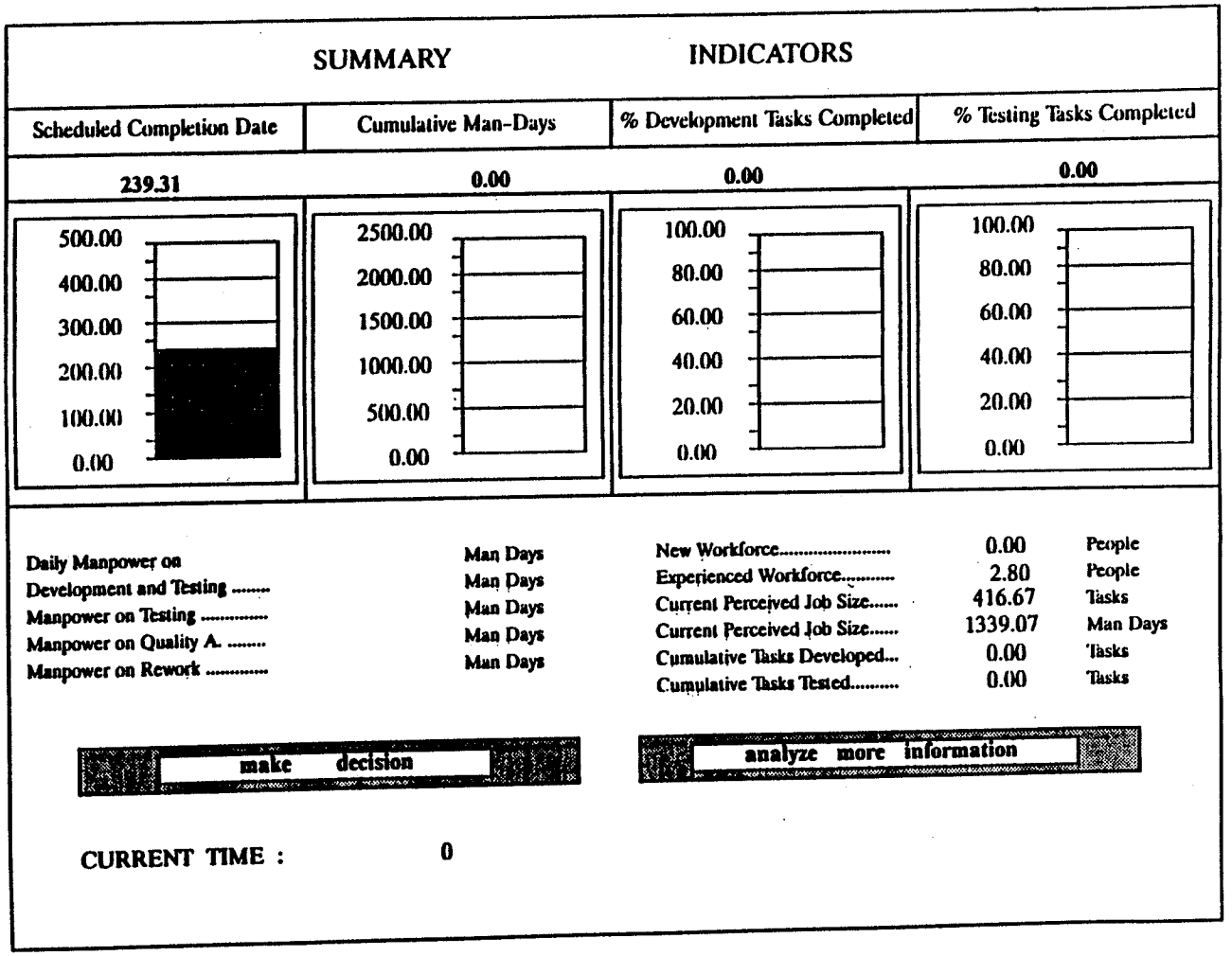

Figure 2. Main Game Screen

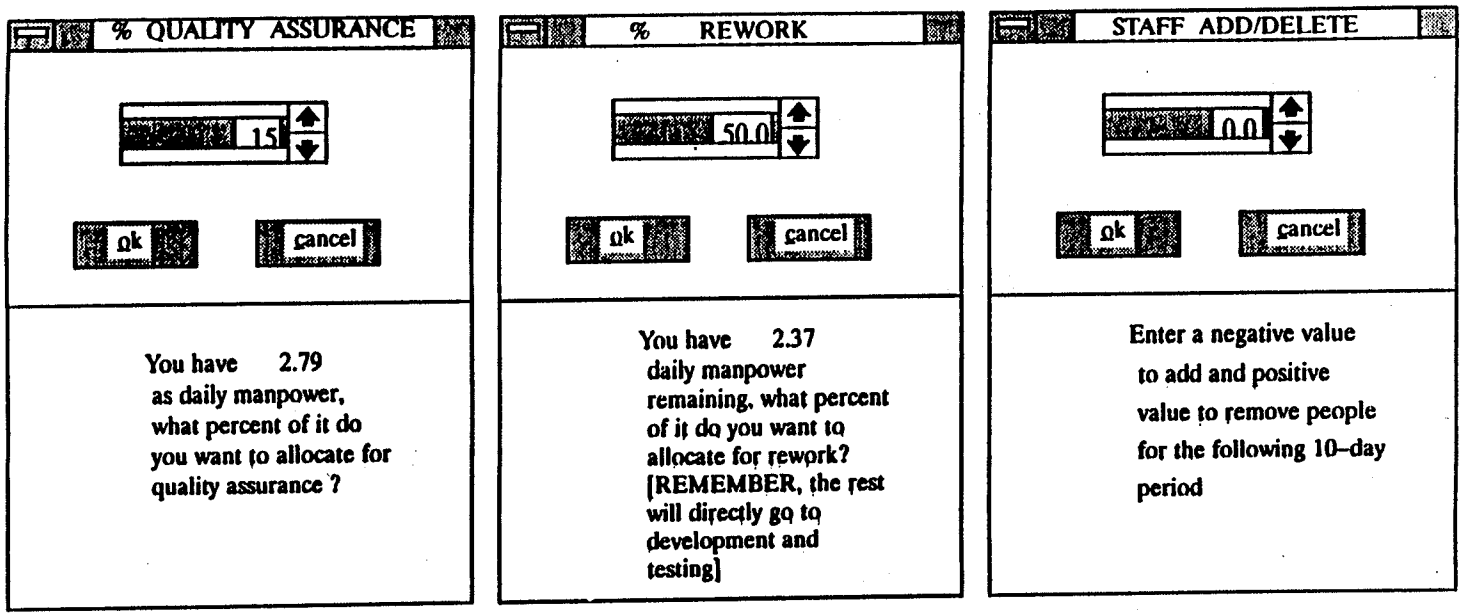

Figure 3. The Three Decision Input Windows

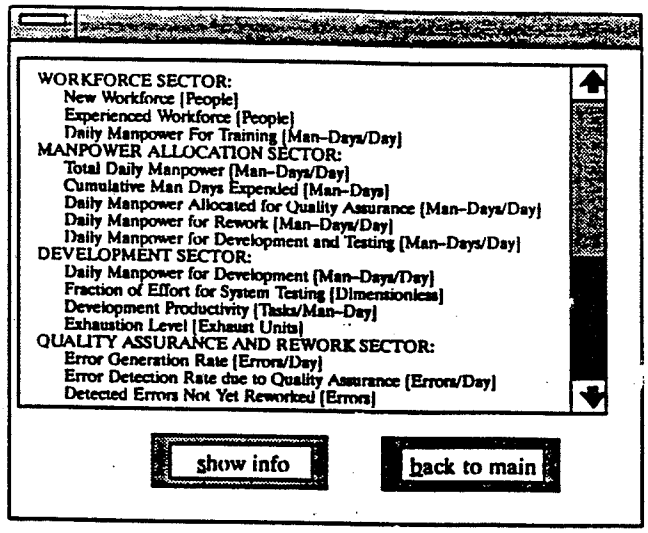

Figure 4. Information Selection Window

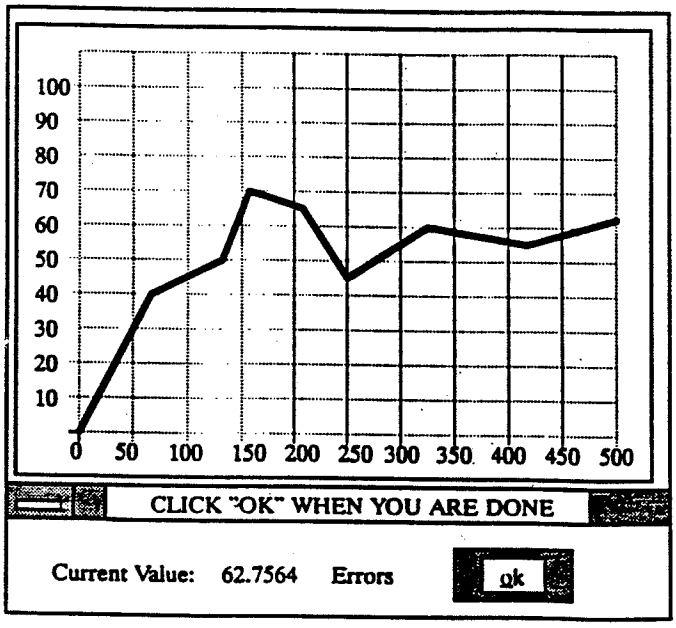

Figure 5. An illustration of graphical information obtained, once a variable is selected.

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	270.36
JBSZMD (Total Job Size In Man Days)	1529.81	2039.70	2129.30	2267.36	1635.68	2486.81	1493.47	2055.40	1339.06
CUMMD (Cumulative Man-Days Expended)	1530.80	2045.44	2127.93	2276.02	1640.11	2487.98	1494.05	2060.52	1337.38
PJBSZ (Currently Perceived Job Size)	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67
CMTKDV (Cumulative Tasks Developed)	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67	416.67
DMPDVT (Daily Manpower For Developemnt/Testing)	4.84	9.30	1.22	10.11	2.85	5.55	2.17	6.88	4.54
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 1. Summary of Performances of Eight Players and the Performance of the Model.

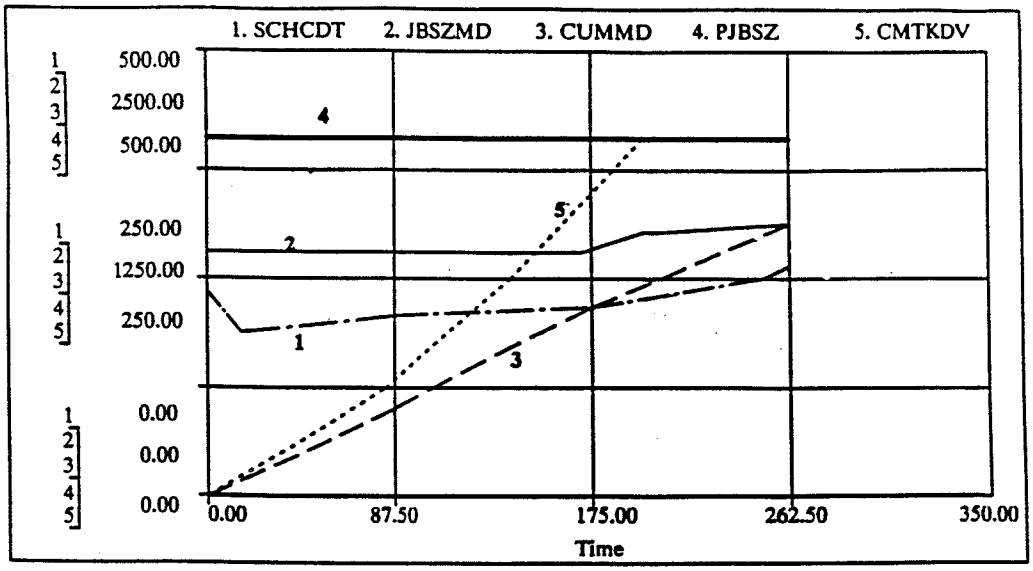

Figure 6a. Behavior Patterns Generated by Player 1.

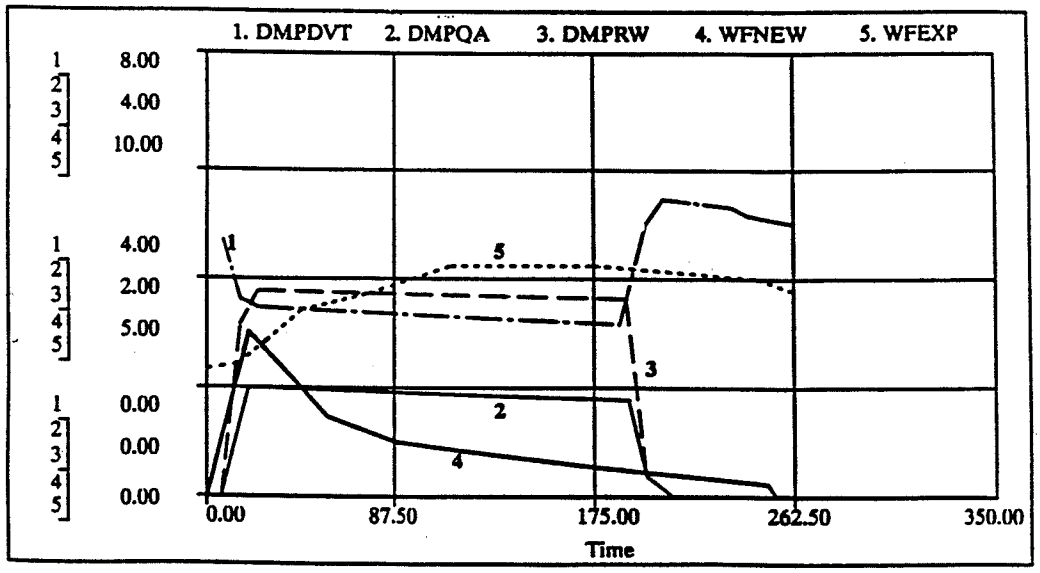

Figure 6b. Dynamics of Decisions Made by Player 1.