Developments In Business Simulation & Experiential Exercises, Volume 23, 1996

DO YOUR STUDENTS REALLY READ THE MANUAL? A COMPUTERIZED CONTEXTUAL DIAGNOSTIC TOOL AS A SURROGATE FOR THE TRADITIONAL STUDENT MANUAL

Peter M. Markulis, State University of New York, College at Geneseo
Daniel R. Strang, State University of New York, College at Geneseo

ABSTRACT

This paper describes how a contextual diagnostic software program developed by the authors can be used to help students learn and better understand the decision complexities, which are part of a typical simulation. The authors developed the program using the multi-media authoring package called ToolBook. The paper presents the program from the perspective of the instructor, the user, and the developer. The authors feel the program they developed can serve as a model or prototype for a variety of pedagogical uses.

INTRODUCTION

(Scenario 1). Picture this: You are sitting in your office on a typical Wednesday afternoon, early in the Semester. As you finish reviewing your notes for tomorrow's lecture, in walk Susie, Tam and Johnny. They are all bright, diligent students whom you have had in several courses. They are presently in your Simulation Course. They slump down in the office chairs (Johnny, of course, is wearing a baseball cap) and begin to tell you the problems they are having with the simulation. They indicate they are having three problems, which are particularly troublesome:

a. They are in last place,
b. They are losing money,
c. Their firm is experiencing substantial downtime.

You know them well enough to know that they have read the manual and you know their problems, obviously, are interrelated. You are perplexed as to why they can not see this. What do you do?

(Scenario 2). Now picture this: It is Wednesday afternoon on a beautiful Fall day early in the Semester. You walk by the computer lab on the way to the golf course. As you walk by, you stop in for a few moments to see what students are doing. You see Susie, Tam and Johnny (Johnny still has on a baseball cap) sitting in front of a PC with their computer printout from the simulation you are using in your course. You observe that the "team" is just receiving a diagnostic sheet from the PC. They enthusiastically tell you that by working with the simulation's computerized manual (known more technically as the contextual diagnostic software) they have just learned why they are in last place, why they have been incoming losses, why their downtime is so large, and most importantly how there issues interrelate. Eureka!" they say, and tell you that they should be able to greatly improve their standing--now that they understand how things work--by the next round of simulation play.

For many of us, Scenario 1 is all too common:

1. Students are told to read simulation manual,
2. Students are given an introductory lecture on simulation,
3. Students play one or several periods,
4. Students come to your office with a whole host of questions about the simulation, (e.g., how come we are losing money, why are our stockouts so large, are we spending too much on R & D?),
5. You know the answers to many of these questions could be found in the manual.

Why does this happen? What do you usually do?

You usually tell them what's in the manual you tell them to go back and read the manual themselves.

This approach is wanting for several reasons--it frustrates the instructor, and it may place the instructor in the vexatious position of wasting time with students or telling students to do what they were supposed to have already done, i.e., read the manual!

We all know that students do not spend sufficient time reading the manual, while at the same time we also know that in many cases the manuals have a wealth of useful information for the students. On the other hand, it is important to recognize that a student may diligently read 100 pages or so of a players manual for a simulation and not derive much benefit from his/her efforts. The reason for this is the student does not have a *context* for his or her reading. That is to say, one may well need to engage in some simulation play before referring to the manual or refer to the manual *continuously* while playing a typical simulation. Many of us have had the experience in our

lives of putting together some piece of equipment while having to continuously refer to the accompanying manual. If we accept this behavior, as "human nature" should we be surprised when our students approach simulation play in a similar fashion.

The Pedagogical Aid

This paper presents a pedagogical aid, which reverses the traditional order of having to read the manual, followed by playing the simulation. The new approach has students using contextual diagnostic software program as they work through the complexities of the simulation play. This approach stresses the importance of having students use a manual or guidebook to help them understand and apply important management decisions but acknowledges that most students are so anxious to play the game that even if they read the manual, they do so in a cursory fashion and out of context. In the new approach, students will not only have access to computerized summary of many of the guidelines from the manual, but the diagnostic software program will serve as a sort of "tutor," helping the students to understand relationships among the various decision variables. An example of the new order of simulation play and how it works is as follows.

Introduction to simulation,
Students play simulation,
Students get output,
Questions arise on various issues and decisions (eg. profit, finished goods, stockouts, downtime),
Student access the diagnostic software,
The software interactively helps them to analyze their problems, and the problem interrelationships, and
Students are provided with a printed diagnosis of their current situation.

The User

A typical example of how this works for "your" team is as follows.

After two periods of play your firm finds itself in this position. You are in last place. You firm has experienced a loss for period 2. The loss arose because you were unable to meet your firm's sales potential. You were unable to meet the sales potential because your firm's production of finished goods was lower than expected which resulted in stockouts for your firm. Your production was lower than expected because your firm experienced an unusually high downtime. Your capital investment expenditures were in the acceptable range (i.e., not too large or too small). Your high downtime was probably a result of an expenditure on maintenance that was too small. Therefore, the problem can be traced backwards as follows.

Low maintenance expenditure caused a large downtime,
Large downtime limited production,
Reduced production caused low sales and large stockouts,
Low sales caused financial losses, and Losses adversely affected your team ranking.

In sum, the contextual diagnostic software program helps the user in the following ways:

It gives pertinent information similar to the written manual,
It responds to problems in a situational context,
It reveals technical details (i.e., game parameters such as inherent elasticities, etc.),
It reveals key relationships among decision areas, and,
It provides a printed diagnosis of the current situation.

An example of the diagnostic screen and computer printout is presented as Figure 1.

FIGURE 1

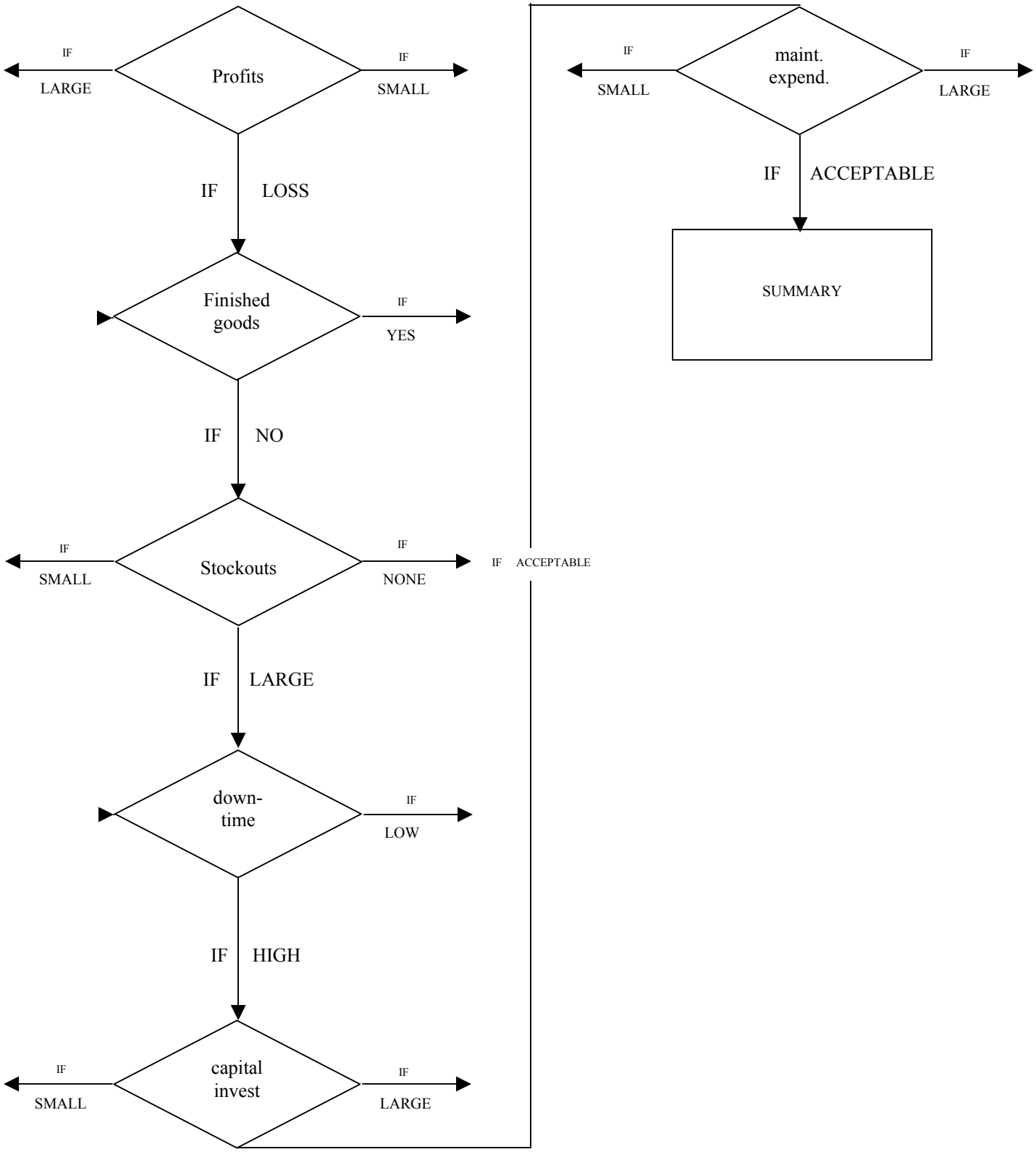
The following is a diagnosis of your current situation. Provides a written summary of the problems (if any) that were discovered in this analysis. Remember hotwords are in bold.	
	Congratulations!! No Major problems were discovered in this analysis.
x	Your firm experienced a finanlcial **loss** for the period.
	Your firm's **Finished Goods Inventory** may be too large. Maybe your marketing mix was not in production.
x	Your firm experienced large **Stockouts**. Your **marketing mix** was too aggressive or you had problems in production.
x	Your **downtime** is above the acceptable level.
	Your **Capital investment** may be too low in comparision to **depreciation**—causing downtime problems.
	Your **Capital investment** may be too high in comparison to **depreciation**—causing downtime problems.
x	Your expenditure on maintenance was too low.
	You have a large **debt** and **marketable securities**. You are borrowing at 10% to earn interest of 6% from securities.
	Your level of **debt** is a matter of concern. Fortunately, you do not have **marketable securities**.
	You have no breakthroughs in **leather** and **rubber wast percentages**. Consider a higher **R&D** expenditure.
	Your **Dividend Policy** is a problem. You should declare $25,000 or 10% of your earnings after tax.
	Print Summary Sheet GO BACK TO START QUIT

It is worth noting that Figure 1 is more than simply a summary sheet of the team's decisions. First, there are twelve rows. If the first row has been checked, that means that the students made acceptable decisions for that period and seem to be on track. On the sheet illustrated in Figure 1, four boxes have been checked. This team apparently lacks some understanding of the issues identified by the check markers. Hotwords (shown in Bold) have been incorporated into the diagnostic screen. The team can simply highlight any one of the "hotwords" in the row containing a check mark and the software will branch to an area which will explain the concepts or concepts pertaining to that decision area and then return to the starting point. This non-linear interaction is one of the ***key advantages*** that the software affords. It is left to the users discretion when to jump and where to go. In this manner the user can map his/her own route through the software and can perform a cursory or an in-depth analysis, which is determined by the users specific needs. For example, the user can determine through a series of interconnected screens that stockouts are related to the marketing mix. This relationship is clarified in the context of the specific game parameters.

The Developer

It is beneficial to understand from the developers perspective what the software is and how it works. Figure 2 presents a flow chart for the example presented above.

FIGURE 2


IF
LARGE
Profits
IF
SMALL
IF LOSS
Finished goods
IF
YES
IF NO
IF
SMALL
Stockouts
IF
NONE
IF LARGE
down-time
IF
LOW
IF HIGH
capital invest
IF
SMALL
IF
LARGE
IF ACCEPTABLE
IF
SMALL
maint. expend.
IF
LARGE
IF ACCEPTABLE
SUMMARY

It would appear that the logic demonstrated in the flow chart could be programmed in many languages. What makes this truly innovative and imaginative in this case, however, is the use of the Multimedia ToolBook software package. ToolBook is one of several hypermedia-authoring packages (see Appendix A for a listing of multimedia-based Information Technology (IT) packages). These packages represent a significant departure from the standard decision support systems (DSS) packages students are often asked to use or develop as part of their simulation learning experience.

Multimedia-based Software Capabilities

Like the standard spreadsheet or database program, most multimedia-based IT packages allow the developer to write programs which can assist the user in making financial, statistical or operational decisions. However, unlike the typical DSS package, the emphasis on multimedia packages is on teaching or providing guidance through the conjoint use of text, video, sound, graphics and animation. Messages are given, stories are told and learning takes place in the arena of multimedia. The developer determines not only the substance of the exercise, but how that exercise should proceed. In the above illustration, for example, the instructor could include sound clips, which accompany the text on financial information. The sound clips might add an auditory explanation to reinforce the text. Or, when explaining the concept of downtime, the developer could include a short series of video clips illustrating the specifics of downtime in an actual setting. In describing various economic concepts, like oligopoly, monopoly, or the demand curve, the developer might have the student actually practice drawing the correct graphs and lines, and then add explanations as to whether the student drew them correctly or incorrectly and why. Iterative quizzes could be added anywhere in the program and would serve to focus the student's attention on key information and areas of deficiency. The program has the capability of using one's own digitized pictures, which can be used when "canned" pictures are not appropriate. Such inclusion also serves to make the program truly unique and may add humor to increase interest levels. For example, the authors have seen a programs in which pictures of the of the instructors looking very academic "pop" up, every so often, to remind the student that the "prof" is omnipresent.

Finally, most of the packages offer the developer the ability to program the software so to suit whatever learning needs and wants he or she feels are important for the users. Below is an explanation of how that is done in Toolbook. (It should be noted that Toolbook is a 'typical' authoring package).

ToolBook programs are very different from the traditional programs with which we are all familiar such as those written in Pascal, Fortran, Basic, etc.
Since ToolBook programs are designed to fully utilize multimedia features, contain numerous computer screens, accommodate "hotwords", and are driven by functional buttons or objects, they do not lend themselves to be reduced to or represented on the written page. Having said that, it may be worthwhile to explain within the inherent limitations of the printed page a little bit more detail of the nature and workings of a typical ToolBook program.

To understand a typical ToolBook program ft is useful to start with the underlying conceptual metaphor, the book and a set of pages and backgrounds within the book. The program is analogous to a book and a number of pages. The user can pass through the pages of the book in many ways. Each page may contain a number of objects (the objects can be buttons, graphics, pictures, etc.). The programmer writes computer code that may give each page, each background, and each object of each page functionality. In this case functionality is a broad, virtually all encompassing term, which means that the programmer defines the result if some action is performed on the page, the background, or object. The combinations and results are also almost limitless.

An example may be helpful. Let's say that page 6 of the program contains a wide variety of objects. One of these objects might be a button, which has a label, YES, on it. The programmer can write computer code, called *script* in the case of ToolBook, which defines what will happen if the user places the cursor on the button, YES, and clicks the left mouse button. This action can set into motion an almost limitless number of responses. To illustrate, it might cause a new screen to pop-up, it might cause a graphic to appear, it might initiate an audio response, or it might cause a transition to a different page. The script for the result is essentially embedded in the object on the page and established by the programmer. Thus, one can not simply provide a listing of the source code, as is typical for other programming languages. In this example, a click of the left mouse button is used to initiate what has been described as virtually limitless responses. For the record, the click of the left mouse button was also illustrative and at the control of the programmer. The programmer could write script that will initiate actions to a left mouse click, a right mouse click, a left mouse double click, etc. These options if not limitless, are so extensive as to preclude a convenient listing here.

Thus, a presentation of all of the intricacies of a ToolBook program-or almost any authoring program- of even modest size is quite difficult to illustrate on the printed page. A fuller understanding and appreciation can only be gained from an exposure to the program itself in an interactive demonstration.

Lessons/Advice To Developers

Finally, for developers, d is important to consider one's strategy carefully before embarking down the road of authoring programs. While we would all like to see scenario 2 above become a reality, there are many considerations to account for before beginning. One's hardware has to be quite powerful. A Pentium machine is almost a necessity as one is likely to reach the limits of the machine on a 486 computer before one reaches the limits of the authoring program. Developers also need to do the following:

Plan their projects,
Prototype the source code (e.g., called ***script*** in ToolBook),
Develop the content,
Author the source code,
Test the source code, and
Prepare the source code for distribution.

CONCLUSION

Scenario 3. It's graduation time and Johnny, Tam and Susie receive their degrees magna cum laude! Tam tells you that he just secured a job with a high flying company because during the job interview he mentioned that he had used a software program which helped him to move from last place to first place in a business simulation. The company was so interested in this that they decided to buy some authoring software and hire him to help write programs to train salespeople. Johnny still has his baseball cap and is off to see the world and Susie....

Clearly, intimidating, coercing or even threatening students to read the manual (or anything else, for that matter) is going to have limited success. But more importantly, even if students read what they were supposed to, it may not be enough to help them understand what they need to know within the context of play. This is the real power of authoring software.

The benefits of using computerized aids to augment or enhance computerized simulations and/or experiential exercises are well documented. From Decision Support Systems (Anderson and Lawton, 1986; Bums and Bush, 1991; Cannon, Carroll and Seamons, 1993; Fritzsche, Rodich and Cotter, 1987; Keys et al., 1986; Markulis and Strang, 1987; Nulsen, Faria and Roussos 1993; Palia, 1991; Rajkumar and Barton, 1989;), Expert Systems (Cannon and Morgan, 1987; Gautschi and Prasad, 1988; Gum and McGregor, 1987; Odom and Murphy, 1992; Sena and Shani, 1992; Sondak and Briggs, 1988; Varanelh et al., 1987), to statistical packages and database packages have all been investigated and frequently touted as facilitating the learning in conjunction with a simulation or experiential exercise. This paper presents the most recent development in pedagogical aids--the multimedia authoring package.

This paper discussed the development and the use of multimedia authoring software program to develop a contextual diagnostics tool for a "typical" management simulation. It is easy for the reader to see that this type of software is not limited to helping one learn about and understand the decision complexities of one's simulation. Such programs could be used for a wide variety of multimedia applications, which can be developed to enhance and enrich the teaming environment.

APPENDIX

Typical Authoring Multimedia Software Packages

Actionmedia is a registered trademark of Intel Corporation
Authorware Professional and *Macrobind Director* are registered trademarks of Macromedia Inc.
Corel Draw is a registered trademark of Corel Corporation
Guide is a registered trademark of OWL International Inc.
IconAuthor is a registered trademark of AimTech Corporation
KnowledgePro, *KPWin*, and *KPWin+ +* are registered trademarks of Knowledge Garden Ltd.
Multimedia Toolbook is a registered trademark of Asymetrix Corporation
Multimedia Viewer, *Video for Windows*, *Visual Basic*, and *Windows* are registered trademarks of Microsoft Corporation.
Picturebook is a registered trademark of Digithust Ltd.
Quicktime and *Hypercard* are registered trademarks of Apple Computer Corporation.
Soundblaster is a registered trademark of Creative Labs Inc.
Turbo Pascal is a registered trademark of Borland International Inc.

REFERENCES

Anderson, Philip A., and Leigh Lawton, "Integrating Personal Computers Into a Course as a Decision Support Tool, Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 13, 1 986, pp. 212-215

Burns, Alvin C., and Ronald F. Bush, "Using DIS 'n DAT as a Decision Support System for a Marketing Simulation Game," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 18, 1991, pp. 5-10.

Cannon, Hugh M., Philip G. Carroll, and Brett L. Seamons, "Using the "IdeaFisher' Idea Generation System as a Decision Support System in Marketing Strategy Courses," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol 20, 1993, pp. 27-30.

Cannon, Hugh M., and Fred Morgan, "An Expert Systems Approach for Teaching Marketing Case Analysis, Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 14, 1987, pp. 31-33.

The Concise Guide to Multimedia. Asymetrix Corporation, 1994.

Fritzsche, David J., Grover W. Rodich and Richard V. Cotter, "Integrating Decision Support Systems and Business Games," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 14, 1987, pp. 63-66.

Gautschi, Theodore F., and Mohad Prasad, "An Initial Step Towards Developing And Using An Expert-System With A Business Simulation," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 15, 1988, pp. 225-230.

Gum, Larry 0. and John D. McGregor, "An Expert System for Financial Planners," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 14, 1987, pp. 80-83.

Keys, J. Bernard, Max Burns, Tom Case, and Robert Wells, "Performance and Attitudinal Affects of a Decision Support Package in a Business Game," Developments In Business Simulations & Experiential Exercises, ABSEL Proceedings, Vol. 13, 1986, pp. 221-226.