Development Of A Computer Simulation Game Using A Reverse Engineering Approach

Ahmet Ozkul, University of New Haven, USA

ABSTRACT

Business simulation games are widely used in the classroom to provide students with experiential learning opportunities on business situations in a dynamic fashion. When properly designed and implemented, the computer simulation game can be a useful educational tool by integrating separate theoretical concepts and demonstrating the nature of actual business decisions. This article presents the author's reverse engineering approach in developing a simulation game for an operations management course. With the ultimate goal of enhancing student learning, the project's objective was to develop a focused game with easy to use tools and controls that could be played in the last three weeks of a semester. Based on an old simulation game, PROSIM III, the new system eliminated rarely used modules and complex rules, and focused on the planning and scheduling aspects of the original game. New analysis tools and attractive screens were added and the simulation process was simplified by employing Internet technologies. Survey responses showed that the new system, PROSYS, was well received by the students. The methodology, findings and experiences presented in this article should be beneficial for other instructors considering similar projects.

Keywords: Business Simulation; Game Development; Operations Management; Reverse Engineering

INTRODUCTION

perations Management is an upper level quantitatively oriented course taught in most undergraduate business programs all over the world. The main teaching method in this course has been traditional and instructivist; based on lectures of theoretical concepts and hands-on exercises with spreadsheets or other software in a computer lab (Raiszadeh and Ettkin, 1989; Hammond, Hartman and Brown, 1996). However, issues with this traditional approach led to the inclusion of new educational techniques to the curriculum based on experiential learning (Kolb, 1984), such as discovery learning (Mukherjee, 2002), problem-based learning (Kanet and Barut, 2003), team projects and collaborative learning (Ahire, 2001), case studies (Morris, 1997), online learning (Arbaugh, et al., 2010), and simulation games (Wolfe, 1997). Computer simulation games can be particularly useful in demonstrating links between concepts, complexities of real life business decisions and their implications in a dynamic environment. Furthermore, a competitive game provides excellent motivation for students to apply their skills and knowledge in a collaborative and cooperative fashion (Garris, Ahlers and Driskell, 2002; Faria, Hutchinson, Wellington and Gold, 2009). Also, opportunities exist for instructors to use simulations as instruments for assessing student learning (Neely and Tucker, 2012). Knowing this, many business management instructors made simulation games an integral part of their teaching, and reported evidence of teaching effectiveness and enhancement of student learning by the use of the simulation (Ricci, Salas and Cannon-Bowers, 1996; Yazici, 2007; Ncube, 2010; Santos et al., 2011; Jeong and Hong, 2011).

For the cited potential benefits, the author has adopted a simulation game for his operations management course, called PROSIM III (Chu, Hottenstein and Greenlaw, 1996) since the early 2000s. Unfortunately, although the software was one of the most popular in its time, it is no longer state of the art in terms of content and instructional technology. The original developers of the system have not updated nor supported the software since

1997. Considering the educational benefits of the existing game, we decided to modify the old game rather than switching to a totally new simulation game. The objective of this project was to develop a customized and more focused version of the PROSIM III game using the latest Excel and Internet technologies. However, as Batley (1991) noted, producing a computer based simulation game was a challenging task for a typical management instructor and thus, a systematic approach had to be followed identifying project's scope, objectives, resources and tasks to do.

In the next sections, we present the methodology used to develop the game, details of the modules developed, and implementation results obtained from the students. A discussion follows providing recommendations for instructors or game developers who would embark on a similar endeavour.

PROSIM III

The first edition of PROSIM was developed for IBM mainframe computers and used in a variety of undergraduate and graduate courses since the 1970s. PROSIM III was developed in the 1990s by Chu, Hottenstein and Greenlaw to provide students with a planned learning experience in dealing with many problems encountered in production and operations management. Adopted by many instructors, its popularity stemmed from its coverage of a wide variety of operations management decision making areas, familiar Excel user interface for decision making, and relative ease of use by the students and instructors.

In the game, a small manufacturing plant is simulated in the computer. In a typical simulated week, student teams review a report showing current plant conditions (e.g. inventory levels, worker proficiencies, product quality levels) and make operational decisions, such as: expense levels on quality and maintenance, workforce (workstation operators) selection, monthly master production plan, production workstation schedules and order quantities and timings for material and parts. All of these decisions must be made by each team every period (week) by using a decision form.

The data in the decision form is recorded on a floppy disk by the students and given to the instructor. Then, the instructor runs the disks on his/her main computer, obtaining a performance report for each team resulting from the decisions. The report shows how much money was spent, how many units of parts and finished goods were manufactured, material and part purchases, and productive and unproductive times. This report is used as a starting point in the next round. The game is played for a number of simulated weeks, course time permitting. At the end of the predetermined length, the game is terminated by the instructor and team performance summary data is generated for grading. The team performance is calculated by a composite measure of delivery percentages and total cost. In the last class, teams reflect on the game experience by a debriefing session, and are asked to identify where they made mistakes or played very well. Although the game was designed to simulate many periods, we used the game in the last three weeks of the semester for 8 simulated weeks after all of the essentials and textbook chapters were covered.

Issues and limitations with PROSIM III

The shortcomings of the current game became more apparent to the instructors over the years. In general, the complaints included the following four points: First, it took a very long time for students to understand the logic of the game. Most of the time, they struggled with the basic transactions of the simulation, rather than seeing the big picture and developing production strategies. Once they had a better understanding, it was usually too late, the game was over. Second, instructors were overwhelmed by the students when they became confused during the simulation. Students needed some help, clarification and direction continuously about the user interface and transactions. Third, the simulation system was designed to illustrate a wide variety of production decisions, normally demanding a significant time commitment. However, in our case, we had only six class sessions for the game, during which only a few major modules could be effectively played. The rest of the modules were rarely used or quickly skipped without much analysis or discussion. Fourth, at the end of each simulated week, decisions were recorded on a floppy disk. If anything went wrong with the disks, or students made a mistake and wanted to reverse it, the instructor needed to enter student data manually and ran the program again. If errors were made for more than one period in a row, the data was irreversibly lost.

To solve these issues and overcome the limitations, the faculty decided to upgrade the existing game, or switch to a totally new game.

Options to acquire a new game

We were faced with the following three options: 1) Abandon the current software and adopt a new simulation game available commercially, 2) develop a brand new game in-house, or 3) upgrade/modify the existing game to fix the complaints. Three faculty members teaching the course evaluated each option. The first option was obviously an attractive one, since the software was already available and ready to use off the shelf. The flip side was that the instructors needed to learn the new software, which included new content, new game rules and classroom implementation details, as well as new material to discuss. When this option was chosen, the experiences, skills and knowledge with the old simulation would become practically useless. Yet, some of the issues, such as the focus on specific learning outcomes and playability within a short time frame, might still remain unresolved unless the new software could be flexible enough to be customized according to our students' needs and course requirements.

In the second and the third option, there were several challenges for the development team, including need for people with domain expertise (operations management and production planning), advanced skills in IT, Excel and computer programming, and knowledge and experience in developing simulation games. In addition to this, with the modification option, understanding somebody else's programming logic and coding was in itself a challenging task. However, if the project was completed in a timely manner and resulted in an efficient, customized product, it might have greatly enhanced student learning. Since no faculty member had any previous game development experience, we decided to modify the old game rather than attempting to develop a brand new game from scratch.

When contacted, the publisher of the game informed us that they did not support the software anymore and were unable to help in the development of a customized version of the game. Since decision support worksheets in Excel used in the game were password protected and the game engine was a compiled executable program without a source code, we decided to use a *reverse engineering* approach to develop our own version of the game.

THE REVERSE ENGINEERING APPROACH

Eilam (2007) defines reverse engineering as "the process of extracting the knowledge or design blueprints from anything man made". Chikofsky and Cross (1990) consider reverse engineering as the process of examining system's components and their interrelationships, and then creating representations of the system in another form or at a higher level of abstraction.

Traditionally, reverse engineering has been applied on physical products, such as machine parts, in order to uncover the secrets of their design, usually in the form of a CAD/CAM model (Raja and Fernandez, 2008; Abella, Daschbach and McNichols, 1994). Raja and Fernandez (2008) provide some of the reasons to use a reverse engineering approach:

- When the original manufacturer no longer exists or does not manufacture the product but customers still need to use the product,
- When some unwanted features of the product need to be eliminated; product performance needs to be improved; or the product needs to be customized but the manufacturer is not willing or unable to do so,
- When rival products need to be analysed in order to design a better product,
- When information needs to be generated on undocumented features of the product.

In terms of software products, reverse engineering helps the designers understand the software's structure, way of operation and its behaviour. However, software reverse engineering is a complex area in software engineering. This is mostly due to the fact that the analyst has to work with executable machine code, from which the original source code is to be generated (Favre, 2010). Application areas for software reverse engineering include: understanding program code like viruses and cryptographic algorithms, discovering flaws and faults with existing systems, discovering unauthorized code in others' software, learning programming and design techniques from

others' products, and discovering opportunities original developers did not think of (Eilam, 2007). For a comprehensive review of the software reverse engineering, the reader is referred to Lu et al. (2002).

Chikofsky and Cross (1990) proposed a life cycle approach that included aspects of "forward" engineering, reverse engineering, and reengineering. Forward engineering is the traditional product development approach, in which each step is executed sequentially, including: determining requirements (objectives and constraints), product design (specifications of the solution), and implementation (coding, testing and delivery of the finished product). As opposed to the forward engineering, reverse engineering starts with a finished product and goes backward to recover product design specifications using "design recovery". The design recovery process is based on not only examining existing documentation and executable code, but also uses reverse engineer's own personal experiences and knowledge on the problem. The aim is to fully understand what a program does. Finally, reengineering is deployed to alter the system and reconstitute it in a new form (Chikofsky and Cross, 1990).

To the best of our knowledge, most of the literature on developing simulation games is based on a forward engineering methodology, which includes designing the simulation, coding the software, documentation and administering the simulation experience (Garris, Ahlers and Driskell, 2002; Lynch and Tunstall, 2008;Thavikulwat, 2004).

Since traditional approach is not appropriate in our case due to our lack of game development experience and our desire to isolate and capture the best features of the old game, we devised a specific plan based on reverse engineering principles (Chikofsky and Cross, 1990) to recover the original design of the game:

- List documented features and rules on the internal game logic by reviewing the game's user manual.
- Identify undocumented features by analysing the game itself.
- Discuss other features and behaviours that are not recovered by the previous steps and develop an equivalent design for that part of the game.

We expected to have some difficulty with the undocumented, hard to understand internal parts of the game. However, the good news was, we did not need all of the game modules, and our objective was to develop a focused version of the game rather than an exact replica of the original. This objective reduced the project's scope and increased its chances for success.

When the original design specifications were discovered, the next step would be to alter the original, and develop a new system.

Design recovery

To recover the design specifications of the original PROSIM III, we first studied the user's manual, and identified major modules, their relationships and variables.

The game creates a business environment where students are challenged with making a number of interrelated decisions each period in order to satisfy customer demand at the lowest possible manufacturing cost. The number of production machines, workers/their skills, quality issues and machine breakdowns limit the output and force the students to make optimal decisions in their forecasting, production scheduling, holding inventories and ordering parts, worker proficiency training, expenses on plant maintenance and quality improvement.

One thing that is clearly noticeable from the user's manual, is the overwhelming details needed just to play the game. Students must read the user manual a few times and memorize all the rules and specific cost figures before they start the game. These rules and parameters should be known by each student in order to play and perform well in the game. To reinforce this reading and memorization process, there is a quiz at the end of the manual. Using this quiz, students are graded for preparedness before starting the game. The original developers' intention in using so much detail in the game was probably to create a realistic simulation environment covering many operations management decision making areas playable for the entire semester in both undergrad and graduate courses. Unfortunately, in our case, we did not have a semester's time for the simulation, or student patience to study and

learn every single detail in the game. In our opinion, students were bogged down in the details, and eventually the extent of the details became a barrier to mastering the game, making rational decisions and achieving learning objectives for an undergrad course.

This "too much details and complex rules" problem seems to be a common issue in most business games, and is also noted by Wolfe and Castroviovanni (2006) in using their business game. They observed that their students made many technical errors and did not use the decision-making analysis and tools sufficiently. Instead, they spent most of their time learning the game's rules. For this reason, we decided, at this stage of the design recovery, to skip the extreme details and complex rules and use only what was essential for a game focused on production planning aspects of the operations management field. This decision is consistent with the findings of Thavikulwat and Pillutla (2010), which suggested that simulations "should involve simple rules that require less time to learn". Also, complexity and amount of the details in a simulation game (to be learned and used in a short time) might be the reason for conflicting results on the effectiveness of simulation as a learning tool found by researchers (such as Greenlaw and Wyman, 1973; Chapman and Sorge, 1999; Anderson and Lawton, 2009; Steenhuis, Grinder and de Bruijn, 2011). In a simulation game setting, students might have difficulty applying their knowledge to a new, changing and unfamiliar situation, and the traditional measures of performance may be measuring only the student team's "ability to interpret in a largely unfamiliar context" (Steenhuis, Grinder and de Bruijn, 2011). For this reason, either more time should be given to the students, or complexity of the game should be reduced to allow them to focus on the actual learning objectives.

During this "design recovery" stage of the reverse engineering, we combined our own knowledge about the game and production management with what we extracted from the user's manual. The manual described many things in sentences, which had to be translated into equations and procedures. Once the basic operating logic of the game became clear to us, we moved on to the next stage to decide what to keep from the old game and which new features to add.

Reengineering the new system: PROSYS

The new system, which we named PROSYS, should address most of the issues encountered in the old system. Specifically, the following enhancements were initially planned:

- The system should use better visuals to attract student attention and improve student comprehension. Colours, charts and graphs should be used.
- Instead of using current complex menu system, all of the tables should be laid over on the worksheets for students to browse easily by using arrows and clicking tabs.
- Historical data (past decisions and their cost and productivity implications) should be kept and easily accessible by students.
- Modules used without much analysis should be eliminated. The decision variables related to these modules should be set constant and provided externally to the students.
- Game details, rules and parameters to be memorized should be reduced to a minimum.
- Key modules should be expanded and detailed.
- Basic manufacturing planning and scheduling logic should not change since it was already designed very well. The new system should still be forecast driven, based on a master production plan (output planning), material requirements planning (MRP) logic, and workstation scheduling. However, these functions should be redesigned to eliminate references to the removed modules.
- Eliminating the need for floppy disks, the system should be linked to the instructor's web site to record students' decision data.
- In some areas in which internal functioning and game logic was not comprehensible, different game logic, parameters, and screen layouts could be used consistent with the new specifications of the game.

Based on these general guidelines, we redesigned student interface of the game, which included the modules shown in Table 1. These modules were used by the students. However, the game engine used by the instructor to process student decisions was a totally different challenge, and is discussed in the next section.

THE SIMULATION ENGINE

The real challenge was actually the re-creation of the simulation engine that processed student decisions and generated weekly reports.

The original game engine was written using out-dated Microsoft Visual Basic 3.0. Since it was compiled, the source code was not available. In the absence of the source code, we took a "black box" approach, of which the internal logic was not known. What we knew or observed was its external behaviour. The new simulation engine should be able to read and save the student decisions, process them according to the game logic, keep records of production quantity and cost implications, and generate a report given to the students.

Table 1. Student modules and enhancements made in the redesigned game.

Module	Enhancement
Forecasting	Student calculated forecasting; columns for specific forecasting techniques, moving averages, and exponential smoothing; MSE, MAD and MAPE
Output Planning	Eliminated confusing numbers and messages; changed its layout and colour scheme
Operators Planning	Eliminated most of the unnecessarily complex analysis; reduced the number of workers to 15 from original 29; provided proficiency values externally in the weekly reports
Product Workstation Scheduling	2 scheduling tables (2 months of play) instead of one; products renamed to avoid confusion; A, B and C for products, and X, Y, and Z for parts instead of X', Y', and Z'.
Material Requirements Planning (MRP)	More emphasis on MRP; students are now asked to create an MRP plan by filling out empty cells; entire 8 week planning data for all products and parts were shown; holding and ordering costs are shown.
Part Workstation Scheduling	2 tables included instead of one; one table for each month
Decisions Submission	No more manual paper submission; no more floppy disks; submission either by email, or by instructor's Web site

We decided to build the new game engine using Excel's Visual Basic for Applications (VBA) instead of a separate programming environment. This way, it was possible to speed up the development process by enhancing Excel with programming functionality. At this stage, the design objectives included processing student decisions accurately, preserving operations and functionality of the original game engine as much as possible, visual display of decisions and progress of the teams, and ease of use by the instructors.

To achieve these objectives, we identified a set of major tasks as listed below:

- Creating a separate worksheet and a special layout and space to hold the entire simulation data for all teams,
- Reading the right part of decision data off of the worksheet cells for a specific team and week given, and writing the results back to the proper place in the worksheet,
- Setting initial values for on-hand inventories, standard production rates, maximum capacities for workstations, actual demands, costs incurred for materials, workstations or workers used,
- Develop special procedures such as forecast updates, worker proficiencies, and defect rates,
- Making the actual production by incorporating impact of worker proficiencies, quality defect rates and randomness,
- A procedure writing all the game results in a formatted, easy to understand report.

Procedure for forecast updates

Since we have years of experience with the game, we had a very good idea of which variable values were within reasonable range. For example, we knew that the customer demand for products X, Y and Z in the first month was around 8500, 6500, and 5000 units respectively. Not knowing these numbers, students had to forecast them before they start the game. After this, the instructor provided forecast updates every week until the delivery.

We decided to use a simple method for weekly forecast updates, in which forecast errors became smaller as the delivery due date approached, which was the end of the month.

Forecast update = Actual Demand + Error + Randomness

We set the actual demands of products to specific constant values for each month. Since the first month is a kind of transition where students get used to the game, the demand is not set very high and should be easily met by the current capacity of the production system. However, the demand becomes higher in the second month, in which students should manage the capacity carefully and make part and material decisions more intelligently. The error term in the equation decreases gradually as we approach to the end of the month. Finally, a positive or negative random number from a uniform distribution, changing from team to team, is added to the forecast, resulting in a slightly varying forecast update for each team. The report given before starting the second month shows actual demand realized in the first month (with no error and randomness). Student performance for that month is judged by a composite measure based on Units Delivered/Actual Demand, and costs incurred.

Procedure for generating worker proficiencies

Worker proficiencies change on a weekly basis and are listed in a table in the weekly report. The proficiency values are generated in advance for an 8 week simulation using linear or nonlinear functions with different starting values, to reflect several scenarios, such as: 1) low proficiency worker with fast learning ability, 2) low proficiency worker with normal learning ability, 3) high proficiency worker with normal learning ability, 4) high starting proficiency worker with fast learning ability. These combinations are used for all 15 workers.

Making the actual production

The procedure first gets the current production rate for manufacturing a product using a given operator at a given week. Then, it calculates what can be manufactured (productive hours) out of the target scheduled hours subject to part X, part Y, and Part Z availability in the inventory. Parts manufacturing is simulated first, and then the product manufacturing comes next in a given week. All scheduled production is executed according to students' work orders written in the weekly decision data, subject to given defectives percentage, worker proficiency levels, and inventory availability. These constraints should be factored in by students in their analysis.

Worksheet layout to record and read decisions data

In response to instructors' complaints about lack of control over team decisions and performance data, we designed the engine's layout to be large enough to keep the data of 14 teams for 8 simulated weeks (as shown in Figure 1). Each team's data, including workstation ID, product/part name, production hours ordered, ID of the worker, and part or material purchase quantities (if any) are kept in a matrix. Also, a summary table on a separate location holds the weekly delivery and cost performance of each team. On top of the worksheet, the instructor enters team ID, and the week number to process, and clicks the RUN button. The VBA script is activated, collecting decision variables, running the simulation and writing the output values to the report worksheet.

Initiating the simulation

The game has some constant values, such as inventory holding, ordering and labour costs, and initial settings such as beginning inventories for products, parts and raw material. Most of these values were directly taken from the original game. Since the level of the initial inventories impacts the performance of production system in satisfying the demand, a high level of raw material inventories are set in the beginning to avoid any shortages in the first weeks of the simulation, during which students learn the game as they play.
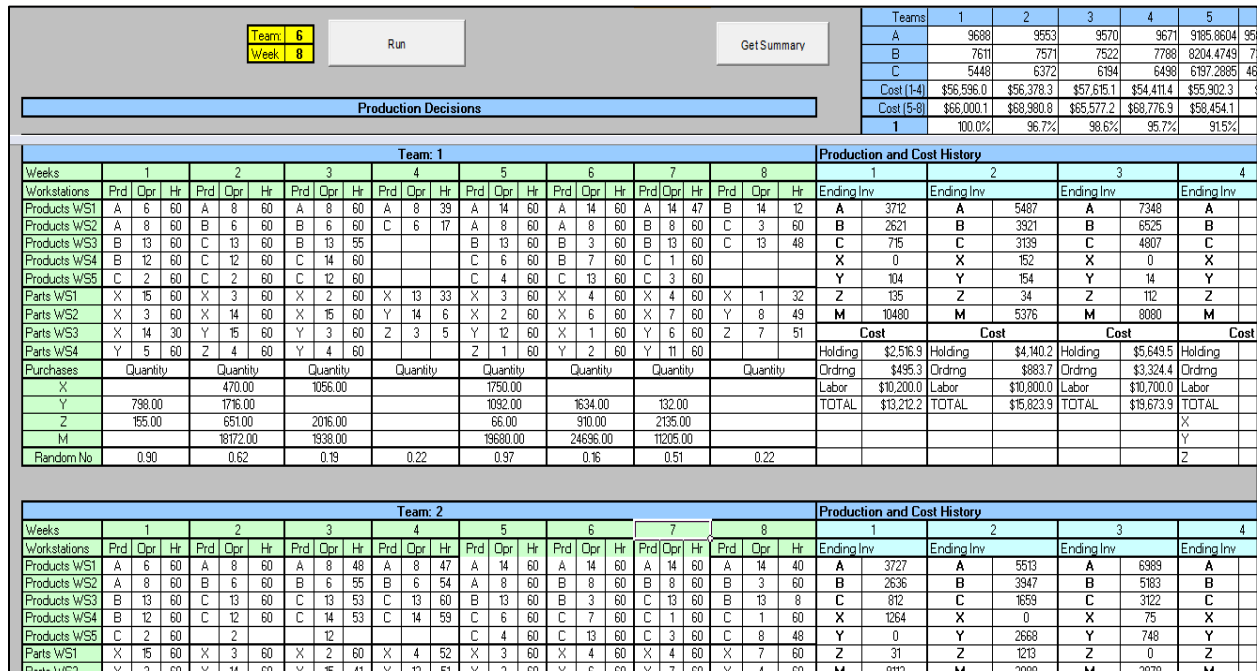
Figure 1: PROSYS simulation engine layout

VALIDATION, IMPLEMENTATION AND FEEDBACK FROM STUDENTS

In terms of internal validity of the simulation, we applied several procedures to make sure the program runs the intended way. First, we printed and examined the code line by line to find out any issues. Second, we ran the simulation with some initial values that their implications are known. For example, we tested the game with zero initial inventories, which gave us raw material and parts shortages verified with a formula using the production rate and the scheduled hours. Also, we put some out-of-range values in the decisions worksheet to see the impact on the simulation. Third, we completed the entire game several times and reviewed the game reports in each period. Finally, we compared the results to those of the original game for consistency. Based on these tests, we made some corrections and other tune-ups until we had sufficient evidence of the validity of the software.

In terms of external validity, our PROSYS is based on core production planning concepts that have direct relevance to the businesses today. For example, forecasting, shop scheduling, and MRP procedures are realistically modelled similar to the actual ERP systems used by companies. Also, many professors all over the world teach these procedures using well-established textbooks.

Student response to the game is very good. Students completed a survey consisting of three questions (shown in Table 2) reflecting their perceptions on the game experience. According to the survey results, students felt that they had a better understanding and integration of the concepts and procedures used in the game. The game was found to be relevant to the real world, and useful for team skills development. Also, in the course's final exam, we asked the students to identify the most significant knowledge/experience gained in the Operations Management. Many students indicated that the most significant learning experience in the course was PROSYS game, and it helped them to integrate many theoretical concepts and demonstrated the impact of a series of decisions on the production performance of a company.

We, the instructors, also observed that most of the student complaints and confusion about the game were eliminated by the new version of the game. Furthermore, we noticed that students were more in control of the game, steering the production system more strategically, and even finding out non-traditional production schedules and other tricks we never thought before.

Table 2. PROSYS Student Survey Responses

Questions (n=80 students) and Scale: Strongly agree (5), Agree (4), Neutral(3), Disagree(2), Strongly disagree (1)	Mean	SD
1) PROSYS helped me to understand and integrate forecasting, MRP and workstation scheduling better	4.3	0.525
2) I believe PROSYS planning is similar to the real planning in actual firms	4.1	0.599
3) PROSYS helped me to understand team dynamics and develop my team skills	4.5	0.530

CONCLUSION

A simulation game not only provides experiential learning, but also motivates and engages students by creating a competitive environment. In this article, we presented a simulation game, a reverse engineering methodology used in the development, and student responses to the game. The new system has been in use since 2007. Since the first implementation in the class, several changes have been made, and it is, currently, an integral part of the author's teaching approach. The system provides a state of the art experiential learning environment tailored to the needs of our students, integrates course materials, and facilitates analytic thinking and group processes. The game is now efficiently playable in the last three weeks of a semester, including initial training and warming up. In the future, the author plans to include more advanced and interesting modules, such as supply chain management functionality to demonstrate the dynamics of a larger business system.

Although the simulation game has many benefits, development efforts are demanding for a typical faculty member. Identifying the internal logic of the game, correct equations, parameters and programming procedures are challenging. To overcome the difficulties, Lynch and Tunstall (2008) suggest a team approach in developing simulation games, a collaboration of subject experts from the academia and game developers from the gaming industry. However, with limited financial budgets, relatively small scale games and audience, it may be difficult for a faculty member to create collaborations with gaming professionals. On the other hand, customizing or reengineering an existing game is difficult and may require additional expertise.

Based on lessons learned in this project, we recommend the following for those academicians who consider reversing a game or developing a totally new one from scratch:

- Determine your goals by considering student needs and feedback, industry/business expectations, and examining games available in the market.
- Consider costs and benefits carefully. Material cost (software/hardware) of this type of projects is usually smaller compared to the time commitments of project participants.
- Follow formal design methodologies, like simulation game design (Thavikulwa, 2004), reverse engineering, forward engineering, prototyping, or System Development Life Cycle (SDLC) suggested for developing information systems (Turban and Volonino, 2010).
- Design your game with the right balance of complexity of rules/details, and realism/challenge for students.
- Most project management principles apply. For example, questions of who, what, when, and how long should be answered in an organized and planned fashion.
- Implementation of the finished product can be based on well-known information systems implementations, such as parallel implementation (new game along with the old one), or direct cutover (quit old one for good, use the new one from now on) (Turban and Volonino, 2010).
- Consider game development a continuous process, and make sure you revise and update your game frequently. Take student and peer feedback into account.

AUTHOR INFORMATION

Ahmet Ozkul is an Assistant Professor of Management in the Department of Marketing and Quantitative Analysis at the University of New Haven, USA. His research areas include production planning coordination in the supply chain, the Bullwhip effect, social networks and data mining. He received his PhD from Clemson University, South Carolina. He published on the educational use of technology, supply chain as a social network, and bundling in the supply chain. He is a member of DSI, POMS, and INFORMS. E-mail: aozkul@newhaven.edu Phone: 203-931-4862, Fax: 203-931-6007.

REFERENCES

1. Abella, R., Daschbach, J. and McNichols, R. (1994). Reverse Engineering Applications. *Computers in Industrial Engineering*, 26, (2) 381-385.
2. Ahire, S.L. (2001). Linking Operations Management Students Directly to the Real World. *Interfaces*, 31(5) 104-120.
3. Anderson, P.H. and Lawton, L. (2009). Business Simulations and Cognitive Learning, Developments, Desires and Future Directions. *Simulation and Gaming*, 40(2) 193-216.
4. Arbaugh, J.B., Desai, A., Rau, B. and Sridhar, B.S. (2010). A Review of Research on Online and Blended Learning in the Management Disciplines: 1994–2009. *Organization Management Journal*, 7(1) 39–55.
5. Batley, T.W. (1991). Microcomputer Simulation for Teaching Operations Management. *International Journal of Operations and Production Management*. 11(1) 5-13.
6. Chapman, K.J. and Sorge, C.L. (1999). Can a Simulation Help Achieve Course Objectives? An Exploratory Study Investigating Differences among Instructional Tools. *Journal of Education for Business*, 74(4) 225-230.
7. Chu, C.H., Hottenstein, M.P. and Greenlaw, P.S. (1996). *PROSIM III - User's manual*. New York: McGraw-Hill, 1-24.
8. Eilam, E. (2005). *Reversing: Secrets of reverse engineering*. Indianapolis: Wiley.
9. Favre, L (2010). *Model driven architecture for reverse engineering technologies: Strategic directions and system evolution.* Hershey, PA: IGI Global.
10. Garris, R., Ahlers, R. and Driskell, J. (2002). Games, Motivation and Learning: A Research and Practice Model. *Simulation and Gaming*, 33(4) 441-467.
11. Ingle, K. (1994) *Reverse engineering*. New York: McGraw-Hill.
12. Greenlaw, P.S. and Wyman, F.P. (1973). The Teaching Effectiveness of Games in Collegiate Business Courses. *Simulation and Games*. 4(3) 259-294.
13. Hammond, D. H., Hartman, S. J. and Brown, R. A. (1996). The Match between Undergraduate Academic Instruction and Actual Field Practices in Production/Operations Management, *Journal of Education for Business*, 71(5) 263-266.
14. Jeong, K-Y. and Hong, J-D. (2011). Learning from Online Beer Distribution Simulation Game. *Int. J. Information Systems and Operations Management Education*, 4(2) 179-192.
15. Kanet, J. J. and Barut, M. (2003). Problem-based Learning for Production and Operations Management. *Decision Sciences Journal of Innovative Education*, 1(1) 99-118.
16. Kolb, D. (1984). *Experiential learning: Experience the source of learning and development*, Prentice-Hall, Inc: Englewood Cliffs.
17. Lynch, M. A. and Tunstall, R. J. (2008). When Worlds Collide: Developing Game-Design Partnerships in Universities. *Simulation & Gaming*, 39(3) 379-398.
18. Lu, C-W., Chu, W.C., Chang, C-H., Chung, Y-C., Liu, X. and Yang, H. (2002). Reverse Engineering. *Handbook of Software Engineering and Knowledge Engineering*, ed. Chang, S.K., World Scientific Pub Co Inc.
19. Morris, J.S. (1997). A New Approach to Teaching Production Operations Management in the Business Core Curriculum. *Production and Inventory Management Journal*, 38(2) 42-47.
20. Mukherjee, A. (2002). Improving Student Understanding of Operations Management Techniques through a Rolling Reinforcement Strategy. *Journal of Education for Business*, 77(6) 308-312.
21. Ncube, L.B. (2010). A Simulation of Lean Manufacturing: The Lean Lemonade Tycoon 2. *Simulation & Gaming*, 41(4) 568-586.
22. Neely, P. and Tucker, J. (2012). Using Business Simulations as Authentic Assessment Tools. *American Journal Of Business Education*, (5)4 449-456.
23. Raiszadeh, F. M. E. and Ettkin, L. P. (1989). Production Operations Management (POM) in Academia: Some Causes for Concern. *Production & Inventory Management Journal*, 30(2) 37-40.
24. Raja, V. and Fernandes, K.J. (2008). *Reverse engineering: An industrial perspective.* London: Springer.
25. Ricci, K., Salas, E. and Cannon-Bowers, J. A. (1996). Do Computer-based Games Facilitate Knowledge Acquisition and Retention? *Military Psychology*, 8(2) 295-307.
26. Ruben, B. D. (1999). Simulations, Games, and Experience-based Learning: The Quest for a New Paradigm for Teaching and Learning. *Simulation & Gaming*, 30(4) 498-505.

27.   Santos, J., Romero, R., Arcelus, M. and Errasti, A. (2011). Teaching Theory of Constraints in 10 hours Using Open Source Simulator. *Int. J. Information and Operations Management Education*, 4(1) 69-82.
28.   Steenhuis, H-J., Grinder, B. and de Bruijn, E.J. (2011). Simulations, assessment and student learning. *Int. J. Information and Operations Management Education*, 4(2) 99-121.
29.   Thavikulwat, P. (2004). The Architecture of Computerized Business Gaming Simulations. *Simulation & Gaming*, 35(2) 242-26.
30.   Thavikulwat, P. and Pillutla, S. (2010). A Constructivist Approach to Designing Business Simulations for Strategic Management. *Simulation & Gaming*, 41(2) 208-230.
31.   Turban, E. and Volonino, L. (2010). *Information technology for management: Improving strategic and operational performance*. Wiley.
32.   Wolfe, J. (1987). The Teaching Effectiveness of Games in Collegiate Business Courses: A 1973-
33.   1983 Update. *Simulation and Games,* 16(3) 251-88.
34.   Wolfe, J., and Castroviovanni, G. (2006). Business Games as Strategic Management Laboratories. *Developments in Business Simulation and Experiential Learning*, 33(1) 31-40.
35.   Yazici, H. J. (2006). Simulation Modelling of a Facility Layout in Operations Management Classes. *Simulation & Gaming*, 37(1) 73-78.

<u>**NOTES**</u>