

FACILITATING BUSINESS GAMING SIMULATION MODELING

Tanabu Motonari
Yokohama National University
tanabu@ynu.ac.jp

ABSTRACT

YBG (Yokohama Business Game) is a business gaming support system in which we can develop and/or play our own business games through the Internet and now used in more than twenty Japanese universities. However even though YBG provides an easy built-in model description language for game developers, there still exist some problems such as model validation, model verification, and parameter tuning for realizing our experience or idea as a game. Furthermore it is difficult to transfer one's game development experience to others. In this paper, we look at YBG system overview and history of development and utilization, and address the modeling facilitation problems and examine agent-based modeling approach.

INTRODUCTION

YBG (Yokohama Business Game) is a business gaming support system which has a game model description language and its processing system, and provides game play environment by generation of execution system based on the description. YBG has been used in more than twenty Japanese universities and it was also used in the universities in United States (2006), China (2005 and 2006) and France (2003). YBG is easy to build business games in contrast to other computer programming approach, because YBG

requires only a description focused to game structure written in the business model description language (BMDL). In BMDL, the game model is expressed as a sequence of command sentence which is a composition of command name, pre-defined mathematical functions, mathematical operators, variables and numerical data or strings. Even though YBG provides an easy built-in language, we need to know how to use it and there still exist some problems such as model validation, model verification, and parameter tuning. And it is also difficult to transfer one's game development experience to others. In this paper, we look at YBG system overview and its history then introduce business gaming simulation modeling methodology and approach. The author proposes a simple gaming simulation modeling approach based on the author's developer training experience over several years and the analysis of existing typical YBG-generated business games, and examine how agent-based simulation approach is effective.

HISTORY OF YBG

The business game development and execution system YBG was developed by the author in 2002. YBG consisted of the game generator (GG version 7) system, the game development environment, the game developer management system and the game model management

Figure1
Architecture of GG version 7 system

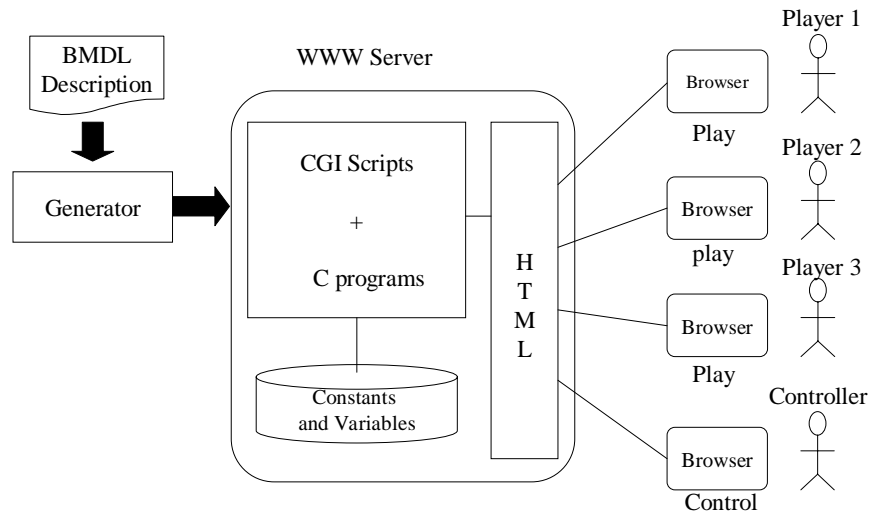
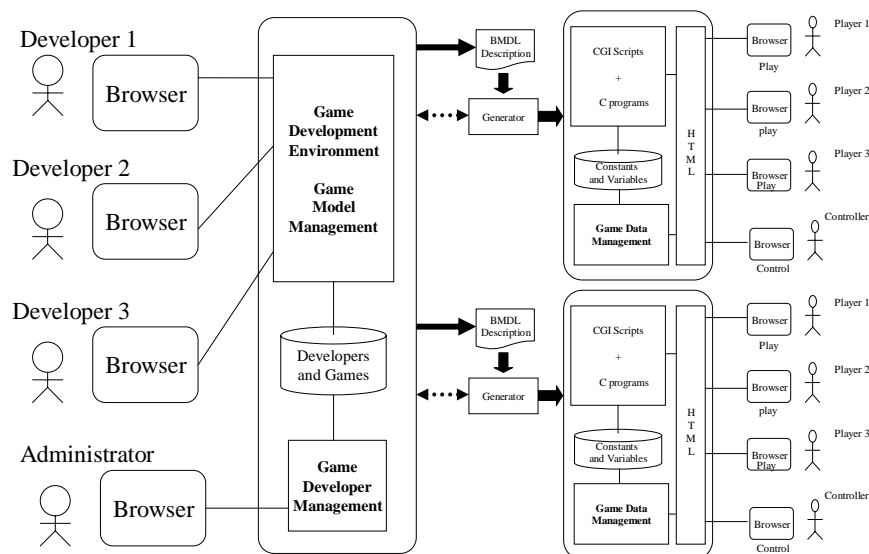


Figure 2
Architecture of YBG1.0



system. The GG system was originally developed by Dr. Yasushi Kuno of University of Tsukuba, Japan (Terano et al. 1999). GG is a simple Perl script which reads a game specification file written in business model description language (BMDL) and dynamically generates business game runtime system components (Fig.1). The generated system can be used immediately with the standard Web browser but GG had to be operated in UNIX shell environment. YBG was developed in order to get rid of the limitation of development environment, and to provide us web interface for game development, game play, and control (Fig.2). The remote development capability of YBG provided many opportunities of developing business games for not only professors but also graduate students, and formed the basis of YBG game developer community. It can be said that YBG enhanced GG's game development support function.

As YBG used in many classes, YBG user as an instructor or teacher required reusability of business games. Sometimes teacher plans to make students play same business game twice. In this case, the first play is considered as a trial, and the result of first play is used for each teams' analysis for the next game play. However in YBG, when teacher gives first play's result online to students and simultaneously make students play the game again, he/she has to generate new game runtime system from the same BMDL file. The author's group had started YBG2.0 development project in order to resolve this problem and improve YBG's original functions. After started the project, the original YBG has been called with version number as YBG1.0. In YBG1.0, data is stored as a pair of variable and its value so it is considered as state of the game. To realize reusability of the generated game, the author introduced "session" concept which means class practically and

implemented it in YBG2.0 (Fig. 3).

In 2004, Yokohama National University's educational practice using business game was selected by the governmental support program for contemporary educational needs and started next YBG development project. First YBG3.0 had developed and tested for one year and then it was modified and became YBG3.1. After several times modification, YBG2007 was finally released in 2007 that is end year of the project. Now it is used in more than twenty universities in Japan. Main feature of YBG2007 is enhancement of BMDL and game facilitation and development, and game players' communication support (Fig. 4).

FACILITATION OF BUSINESS SIMULATION MODELING

In an YBG-generated game, all players basically make decision in the shared economical environment, namely all players have the same role. Players compete with each other for specific game objective which is typically profit maximization or cost minimization. There exist a lot of YBG games and they are used in many universities in Japan at the moment. Moreover, in several classes in graduate school, the students have to create their own business games to acquire deeper understanding of a certain business related topics such as accounting, finance, marketing, production, operation management, enterprise strategy. However, teachers, especially inexperienced teachers, are facing difficulties in teaching business simulation modeling. It is also difficult for inexperienced teachers to create educative new business games, and sometimes they feel uneasy about effectiveness of their game.

Figure 3
YBG2.0 uses one generated runtime system and multiple session data

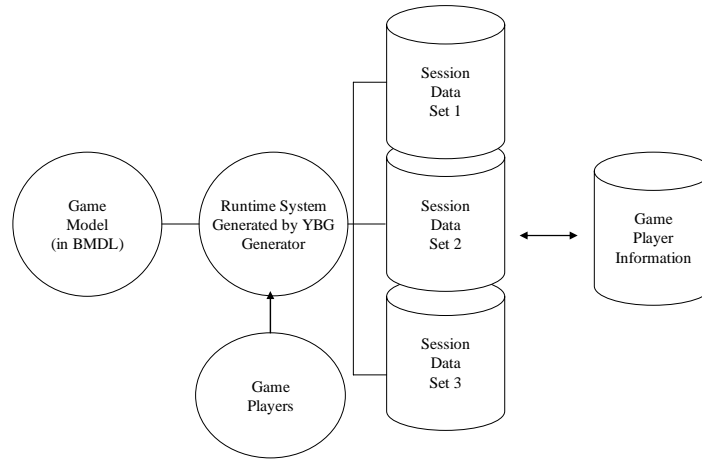
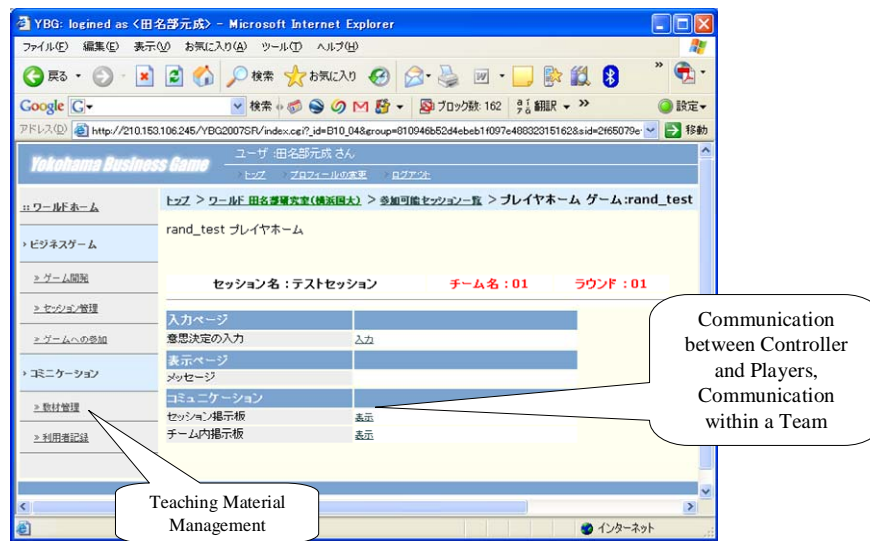


Figure 4
YBG2007 enhanced development support, game facilitation, and players' communication.



For this problem, Hiroaki Shirai, one of the originators of GG/YBG, proposes four-step modeling approach (Shirai, 2001); (1) to recognize own experiences as system object in the real world, (2) to describe its recognition as conceptual model using diagrams, (3) to transform it into mathematical model, and finally (4) to represent as implementation level expression (Fig. 5). Even though graduate student majoring business studies can create their own business games according to the approach in the class, Shirai's four-step approach is generally not so easy that business modeling

beginner can not accomplish without teachers instruction. By extending Shirai's approach, a framework for business game development is proposed (Koshiyama, Terano & Suzuki, 2007). In general, a framework is a brief set of ideas for organizing a thought process about a particular type of thing or situation. A framework helps people by identifying topics that should be considered and by showing how topics are related. The proposed framework has two dimensions; software development stage and idea crystallizing stage (Fig. 6). By using the framework, business game development

Figure 5
Four-step approach to the business gaming modeling proposed by Shirai.

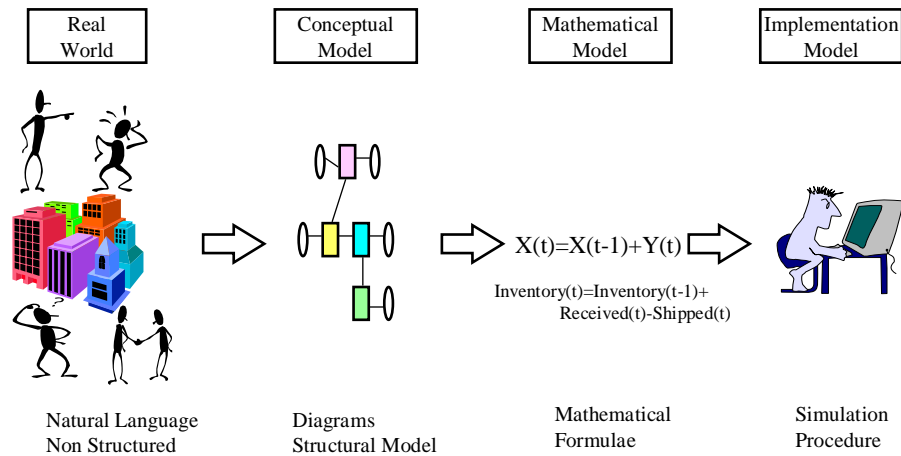

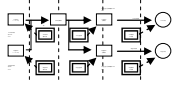


Figure 6
A Framework for business game development (modified by the author)

		STEP1	STEP2	STEP3
		Basic Design	Conceptual Design	Detailed Design
Idea		Essentials of Game -fear of a lot of inventory -seasonal goods selection	Conflicts of game -procurement and stock -seasonal goods and regular goods	Scenario of game -purchase order is limited -Bulk buying causes high discount rate
variables	IN	Basic Level Input items - Purchase order	Conceptual Level Input items -Seasonal Goods (Summer) -Seasonal Goods (Winter)	Numerical Level of relationship between items -Possible Sale =Inventory + Item Received -Ord_Received =Demand * Power
	OUT	Basic Level Output items -number of sales -profit	Conceptual Level Output items -Order received -Surplus	
Expression		Image chart 	Schematic model 	Programs -prop Ord_Received = Demand by Sales_Power; -let Comp_Adv = (2000/Price)+(Items*30)

process is more structured.

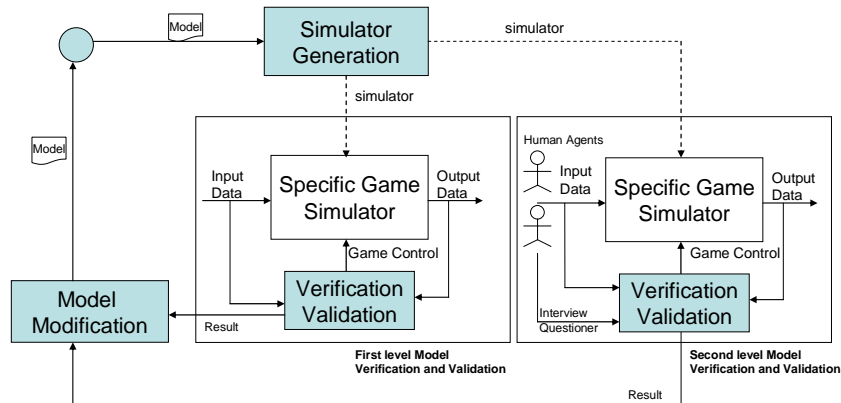
Now we look into how business gaming simulation is actually modeled with YBG based on the author's experiences. Six steps for modeling are shown. The steps introduced here is a business simulation modeling steps in YBG. The term YBG originally means business game generator system, but in this paper we use YBG as a concept of simulation lifecycle in which people examine their ideas or experiences in terms of business structure or dynamics through creating a business game, executing it together with

human and/or machine agents, and rebuilding it based on the analysis of the data generated by the simulator or evaluation by the human agents. In other words, YBG is one of a business game modeling methodology (Fig.7).

GAME OBJECTIVE AND GOAL

In order to realize our idea or experience as a business simulation game, YBG game developer should start with identifying game objective and game goal. We call this stage step 1. The game objective corresponds to a learning

Figure 7
YBG business game modeling cycle.



objective for the players in a class. The game goal is a goal players try to achieve in a game. It is important to distinct objective and goal. “Profit maximization” or “cost minimization” is a typical game goal, but “providing hands-on understanding of the break-even point” is a game objective as an example.

ELEMENTS AND RELATIONSHIP

Next, in step 2, the game developer should identify input, process and output. Input stands for input variables whose value is decided by game player every round. The input variables correspond to a part of elements of a system being modeled. Process means relationships among the elements of the target system, and it is mainly represented as mathematical formulae so that the specific game simulator calculates all the values of variables. However in this step, the developer does not need to specify all the relationships as rigorous mathematical expression, but needs to define simple relationships as combinations of addition, subtraction, multiplication and division of variables, or simple description in natural language. Output means variables whose values are returned to game player. The output variables can be used in process definition, and also can be input variables. The game developer has to identify which variable should be disclosed to all players. The values of output variables are used by the players so that they can analyze other players’ strategy or market mechanism, and evaluate their previous decisions or strategy and make decision for the next round.

INPUT OUTPUT SYSTEM

In step 3, the developer should describe input, process and output in BMDL using a template given by the YBG simulator generator (Fig. 8). The template is a text file and divided into several parts. The first section is a part of “general definition” in which general settings such as name of a game, maximal number of players and maximal number of rounds are defined. The second section is a definition of

the input variables. In this section, input variables, their range, and default values are defined, and a design of the “Input Page” is also given. In next section in the template, team variables are to be declared. All variables and constant, except input variables, must be declared by tvar, sCON, svar or gCON command sentence in advance of usage. The variables can be declared with initial value or values of former (round 0, -1, -2, ...) rounds.

The process is to be defined in next section. Most of the relationships among variables are represented in tlet sentence. When a game controller makes game round forward, the mathematical expressions in tlet sentence are evaluated and all value of variables are calculated. Besides the tlet command we can use aggregation or distribution commands. Aggregation commands (tsum, tmax, and tmin in YBG1.x) are used to calculate summation, maximum value, or minimum value. Distribution commands (prop, pinv and dist) are used to distribute value of some variable to some team variable. For example, pinv command usage is defined as “pinv team_variable = expr1 by expr2;” and value of expr1 is distributed in proportion to inverse value of expr2 to each value of team_variable. The prop command distributes value of some expression in proportion to value of some other expression. The dist command distributes value of some expression in order of values of some other expression.

Most of the YBG-implemented games have the distribution setting in their model description. The distribution design in a simulation such as which value of variables to be distributed or how to distribute is quite important because it may have a great influence on its game balance, so distribution should be carefully designed and also frequently re-designed if needed. The author proposes a semi-structured “process” design method in which developer should design two sub processes; distribution process and team process in the input-output system (Fig. 9).

Figure 8
Structured business game description in BMDL

```

# General Definition
def game-name SimpleGame
def max-team 5
def max-round 10
scon demand 0 1300 1334 1351 1369 1379

# Input
ipage price Decision
<p>Input Selling Price</p>
ivar Selling_Price range 0 1200 700

# Process
tvar NofOrdRecieved
pinv NumOfOrdReceived = demand by SellingPrice;

# Output
opage sales sales_status public
<H1>Sales Status</h1>
<P>${ROUND}: Total Demand: ${demand}</p>
begintable
out teams
out teams-vars SellingPrice NumOrdReceived
endtable

```

Input Definition

Process Definition

Output Definition

After the definition of the process, the developer identifies which value of variable should be fed back to the players, decides which variable should be disclosed to all and which should be individually provided to the teams. For instance sales information such as every team's selling price and number of order received from customers at certain round is usually provided as common information at next round, but balance sheet or income statement is provided individually.

MODEL VERIFICATION

In step 4, the developer should verify the model under development with compile-simulation-modify cycle. The cycle is alike to so called debug in computer programming. The YBG simulator generator helps to detect syntax error and a generated specific game simulator helps to detect logic error and runtime error. To verify a model, the developer has to assign values to the input variables and check the result of calculation from the specific game simulator. If a game is designed for *n* teams and has *m* input variables and *k* internal/external variables, the developer has to set *nm* values for input, and check the result, up to *nk* variables, for each round. This work is time consuming. The author introduces a technique to avoid this problem later in this paper.

MODEL VALIDATION

As stated above, YBG modeling approach is prototyping, verification is prior to validation. But it does not mean validation is less important than verification. Providing an operational model helps the developer to refine his/her idea or experience, and at the same time it helps to validate his/her model. After the simple verification, the developer validates the model (step 5). Validation in YBG is

actually same as verification, but the meaning of these activities are different. In verification, the developer's concern is whether the generated simulator works according to the model, but in validation the developer are interested in how the model reflects his/her idea, or object system. Therefore, in validation, the developer may want to examine typical player's strategies, distribution mechanisms for a certain resource, and the other game specific parameters. Validation is a heavy work as well as verification. To resolve the problem the author proposes the agent-based approach to the simulation modeling.

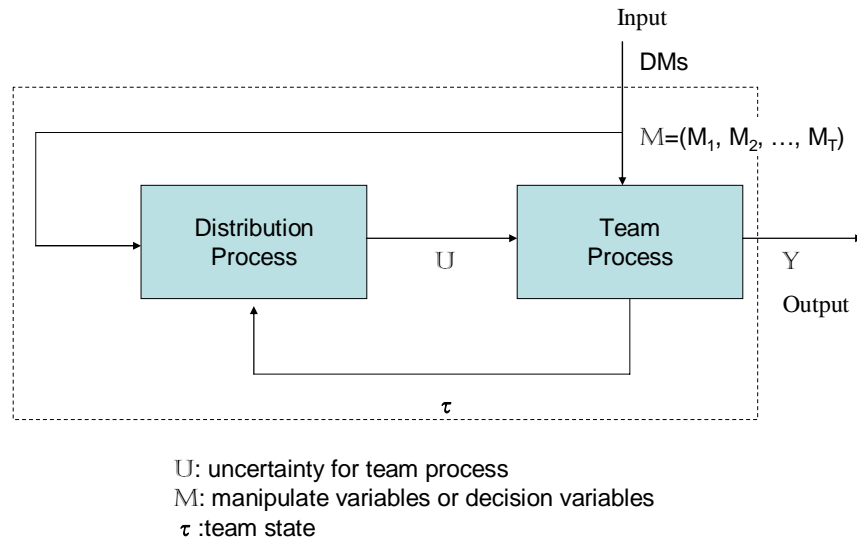
EXPERIMENT WITH HUMAN AGENTS

Final step is to validate the model by experimental game play with human agents. This step is sometimes omitted due to the time constraint. However it is very important to improve quality of the gaming simulation. We can not completely design dynamics of the game due to the nature of gaming simulation, that is, gaming is a simulation in which human acts. Thus the developer should gather quantitative and/or qualitative data from players. Even though enough validation with human agents can not be performed, the result of actual play of the game would facilitate the developer to sophisticate the model.

**AGENT-BASED APPROACH TO
SIMULATION MODELING**

In the simulation modeling with YBG in action, there are two big problems to be resolved, that is, to facilitate verification and validation. In order to facilitate verification, as mentioned above, it is needed to generate values for the input variables automatically. Every YGB generated runtime

Figure 9
Semi-structured process design



system assigns default values to the input variables that values are not submitted yet. The developer can generate values for the input variables using this default assignment function and a simple technique as follows.

```

ivar inputVar1 range min_value max_value default_value
ivar inputVar2 range min_value max_value default_value
ivar inputConfirm select 0 0 No 1 Yes
...
tlet if(inputConfirm == 1) {
    var1 = inputVar1;
    var2 = inputVar2;
}else{
    /* definition for team variables var1 and var2 */
}

```

The `ivar` command declares input variable, and defines a range (`range` option) or a set of values (`select` option) and its default value. The third `ivar` line generates a select button in which player can select “No” or “Yes”, it has a value of 0 or 1 respectively, and default selection is “No”. The `tlet` command executes calculation for every team. The input variables defined by `ivar` command are usually used directly, but in this case input variables are used indirectly, that is, if `inputConfirm==1`, `inputVar1` and `inputVar2` are substituted for `var1` and `var2` respectively, otherwise values of `var1` and `var2` are calculated according to the definition. Since the default value of `inputConfirm` is 0, players need to change its value explicitly. However for the verification, it is useful to automatically generate values for input variables. The developer can define its value as he/she wants. After modeling, the developer only needs to change the default selection setting for `inputConfirm` as “No”, and the game can be used as a released version.

Not only an automatic value generation for input

variables, but also an effective definition of output variables is important. The `opage` command is used for a design which variable should be fed back to the players and controller. The developer may define internal team variables and use them in `opage` definition so that he/she checks their values in output page for controller. The YBG generated runtime system provides output data in CSV format as well as HTML table format. This interface can be used by other software systems for in-depth analysis.

After verification of the model, developer needs to validate it. In business gaming simulation design, validation means to examine how the model works as well as the developer imagines. Typical developers’ interests are systems dynamics and optimal strategy. Model description that is to be considered is divided into environmental setting and individual strategy settings. For an environmental setting, or typically market mechanism, the BMDL provides `prop`, `pinv`, and `dist` commands that distribute some resources to each team. The author newly embedded `topt` and `tspv` commands for individual strategy settings. The general form of `topt` command is “`topt team_variable = expr1 top expr2;`” and a team number of the team whose value of `expr1` takes `expr2`-th place is substituted for `team_variable`. For example,

```
topt maxOrdReceivedT = OrderReceived top 1;
```

gives a team number of the team which received the greatest number of order from the customers. Note that `topt` can be used like `argmax` which is an argument of the maximum of a given function. The `tspv` are used as “`tspv team_variable = expr1 of expr2;`” and a value of `expr1` of team `expr2` is substituted for `team_variable` by following statements.

topt maxOrdReceivedT = OrderReceived top 1;

tspv nextSellingPrice = SellingPrice Of maxOrdReceivedT;

We can allow a specific team to set next selling price as same as one decided by the team which received the greatest order.

Agent is an input-output system that perceives an environment and takes action against the environment, and has a detector, an intelligent mechanism and an effector in itself. Multiagent is a collection of agents. Agents are divided into two categories, i.e., non-learning agents and learning agents. Using the technique stated above and the special commands, we can realize intelligent agents in our business simulation and use them for the model validation or exploration of various strategies under the given settings.

As a practical problem, when almost all the human agents are not active enough, sometimes the game becomes to be boring for player. To avoid such a case, we can use machine agent in a business simulation. In YBG implemented multi-newsboy game which is an extension of the newsboy problem, the author assumed some typical strategies and implemented them in agents, and examined the game balance, and explored an adequate machine agent strategy. This work took half a day. The author made 96 students play this game in ten same setting markets which is to be played in ten players. As a result, the four machine agents took 2nd, 7th, 2nd, 1st place in different market respectively. The agent does not learn, i.e., does not rewrite its rule value, but has a simple strategy. In a sense, this case illustrates effectiveness of agent-based modeling for gaming simulation. The next case is more complicated than the former one. In a game in which player has to decide values of three variables (selling price, number of order for raw material and number of production), agents estimate total demand and assume stochastic distribution and estimate distribution parameters, and decide selling price to maximize the profit. It took also half a day to build the game, but the agent could not have a good record in actual play with 96 human agents.

CONCLUSION

In this paper, some facilitation approaches to business gaming simulation modeling is introduced. Based on the author's experiences, six-step modeling approach, and semi-structured process design method are proposed. Model verification and validation are important for designing a gaming simulation, so the author introduced some special commands which is useful to implement machine agents, and illustrate some usage. The two cases using non-learning machine agents indicate effectiveness of agent-based modeling approach to gaming simulations. The YBG system with agent-based simulation functionality can provide us more powerful modeling tools, and proposed approach should be examined in detail. For further research, the framework with agent-based approach should be applied for education of graduate school level, and some practical problem solving. For the development knowledge transfer

problem, author's YBG development group has started "YBG development seminar" for the instructors in Japan, this activities should be supported by Groupware or SNS combined with YBG in order to form game developer community.

REFERENCES

- Koshiyama, O., Terano, T., Suzuki H. (2007): Proposing a Framework for Business Game Development, *Proceedings of the National Conference of JASMIN 2007 Spring*, Japan Society for Management Information, pp.418-421.
- Shirai, H., Tanabu, M., Terano, T., Kuno, Y., Suzuki, H., Tsuda, K. (2003): Game development toolkit for business people in Japan, *Simulation & Gaming*, Vol.34, No.3, pp.437-446.
- Shirai, H. (2001): *Business Model Creation Technique*, JUSE Press. (in Japanese)
- Tanabu, M. (2006): e-Learning Type of Business Game into Practice, *ICT Katsuyo Kyoiku*, Kaiseisha Press, pp.124-128, 2006. (in Japanese)
- Tanabu, M. (2004): Implementation of Business Game Activity Support System, *Journal of Electronic Science and Technology of China*, Vol.2, No.3, pp.27-32.
- Terano, T., Suzuki, H., Kuno, Y., Fujimori, H., Shirai, H., Nishio, C., Ogura, N., Takahashi, M. (1999): Understanding Your Business through Home-Made Simulator Development, *Developments in Business Simulation and Experiential Learning*, Vol. 26, pp. 65-71.