US-China Education Review A, February 2018, Vol. 8, No. 2, 61-72
doi: 10.17265/2161-623X/2018.02.003

Mapping and Simulation of Educational Games Using a State Transition Diagram and a Rules Base*

Sandrerley Ramos Pires, Tobias Gonçalves Pires
Goiás Federal University, Goiânia, Brazil
Dulcinéia Gonçalves F. Pires
Goiás Federal Institute, Anápolis, Brazil

When educators are planning their classes, one of most challenge tasks is to awaken students' interest for the lesson content. The use of educational games may support teachers to reach this goal. However, this strategy requires high investments. This paper proposes that a formal model and a knowledge base (KB) that can map all necessary details of an educational game, allowing the simulation in a computational environment. Also, this work shows a software able to interpret that formal model and to understand the KB by use of an inference engine. With these resources, the software simulates an educational game, offering students it on the Internet. The first step for the game creation starts with the manufacture of the formal model and the KB. This paper uses a state transition diagram (STD), which can map all possible paths in a specific game. This model mapped the dynamic of several educational games tested. Besides the STD, we used a KB to store rules used in the game control. After mapping the game, teachers must submit the model to the built program. The program reads the model and simulates the mapped game in a computational environment. It has an inference engine that controls the games' flow by the various states of the game. The start stage is equivalent to the initial state of the STD. Users may pass to the next stage only if they fulfill the game rules. The final state of the STD is equivalent to the final stage of the game. To show a feasibility of the proposed approach, some educational games were developed and used in practical classes. The game creation is a hard task, and however, teachers may improve it indefinitely. In addition, teachers may exchange their games between them, creating a gaming base in their institution. The work conclude that objectives were reached, due to the proposed model can map all educational game figured out by participating teachers and the software is able to simulate all the games mapped in the proposed formal model.

Keywords: educational games, expert systems, game simulation, process modeling

Introduction

Games are simulation of small situations from the real or imaginary world. Players are participants from this simulated context, where they act to get a better success level with theirs moves in the game. The players' moves aim to reach the winner state of the game. Games is frequent used in business and academic world. This

* **Acknowledgement**: The authors would like to thank to Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) by financial support to project which originate this work.
Sandrerley Ramos Pires, Dr., associate professor, Electrical and Computer Engineering School, Goiás Federal University.
Tobias Gonçalves Pires, Master's degree student, Electrical and Computer Engineering School, Goiás Federal University.
Dulcinéia Gonçalves F. Pires, M.D., lecturer, Logistics Department, Anápolis Campus, Goiás Federal Institute.

didactical resource allows students to live real situations from the business point of view. In this environment, students (players) analyze a context and make decisions to reach a specific objective.

Institutions and companies are increasing the use of business games because this strategy improves the teaching and learning process (Vicente, 2000). In the academic area, there is a consensus that by using this sort of games teachers may achieve better results in the learning process. The union of theoretical and practical contends aid the students to assimilate the course contents. In companies, the employees can train some problem situation looking for a good solution. Employers may analyze their behavior when they are faced with complex situations.

In Brazilian market, there are many didactical games available as the Bernard (2017), OGG Simulação Empresarial (2016), and the Simulare (2017). All of them attend courses in the business area. Although these mentioned business games are good options to academic use, some considerations are important to be enumerated:

1. Several analyzed games are inflexible to insert new features. They do not allow to insert new rules neither new cases;

2. Some games are not available to play on the Internet. This missing feature reduces the game's awareness. The use of games out of classroom and in e-learning must be an eliminate criteria to choose a didactic game;

3. Games in the market normally are expensive. Besides, some games require additional training for teacher, increasing more their prices.

This work bases on those restrictions to propose a computational system to build and simulate didactic games. Teachers may create their own games, increasing the available resources to their courses. Nowadays, the use of didactic games occurs more in business courses. With this proposed system, the teacher can use didactic games in any kind of courses. Teachers build a didactic game mapping it in a formal model. Next sections show how to map a game.

All games analyzed in this work, as Bernard (2017), OGG Simulação Empresarial (2016), and Simulare (2017) are simulation of some organizational process (Havey, 2005). The formal model used in this paper can map all kinds of process. This model is used to create a computational software able to assimilate game's information and simulate it. Besides, the computational software let the game be available on the Internet.

The basic hypothesis of this work is the use of a formal model able to map the dynamic of a game, as well as its rules and restrictions, turning it possible the simulation of this game in a computational environment. This work also aims to allow that a teacher creates its owns didactical games using the implemented software.

The paper relates the executed tasks to determine what the game's meta-information are necessary to permit its computational simulation. The second section shows some basic fundaments to aid to reach a good reading and to understand of the proposed approach. The third section enumerates the requirements to formal model maps a game, as well as, the proposed approach. The fourth section shows the obtained results, where by an example of how to create a game in the software, by doing this, readers may know the implemented interface. Finally, in the last section shows the work conclusions.

Fundaments

This section shows some essential aspects to understand the proposed approach. It begins with a general vision of business game. To map an organizational process, this work uses the state transition diagram (STD) (Wagner et al., 2006). The STD have enough semantic to represent all dynamic portion of an organizational

process. To map the rules of the game, this work presents a brief vision of the expert systems (Russel & Norvig, 2013). Since each step forward in the game needs to satisfy the game rules, this work uses a knowledge base (KB) to map these rules. This base is like KBs used by expert systems.

Business Game

The courses from the business area are the greatest users of this kind of game. The students compete between them, normally, in the computer laboratory of school. These games permit to visualize the partial scores, turning the competition livelier and increasing the quality of results. From the results of a game, it is possible to analyze the strengths and weakness of each student, permitting corrections of its performance. This way, the games improve the effectiveness of a business courses.

The American universities use the business games as didactic alternative since 1950 decade. In Brazil, the companies began to use the business games in beginning of 1960 decade (Lopes & Souza, 2004). But the disseminated use of business games just happened during the 1980 decade. This dissemination occurred in companies and in universities.

Gramigna (1993) classified the didactic games as behavior games, process games, and market games. The behavior games work with cooperation, flexibility, and courtesy. In the process game, the emphases are the technical abilities to negotiations, how to lead groups, set up strategies, and manage finances. Finally, the market games have the same features of the process games, but they have another focus, such as competitions, market research, supplier relationships, and outsourcing.

Sauaia (1995) proposed different educational objectives to each academic level. In the undergraduate level, the games aim to create a systemic vision of the organizations, develop the critical spirit, and stimulate learning. These abilities are important to develop a skill of manager. In postgraduate studies, the students use games to apply their abilities in a practical context and to interact with others to discuss their actions and decisions.

This use of didactical games brings many advantages to teachers. But the high cost to buy the games makes it difficult to disseminate its use. The proposed environment solves this problem because teachers can create their own games and maximize the use of didactic game in their courses.

State Transition Diagram

This model is used in the software engineering since 1960 decade (Sommerville, 2003). The STD represents the dynamic behavior of a system, mapping all possible states a system can be. Besides, the STD also maps the actions which cause the changing of state.

Figure 1 presents the elements of a STD: initial state, final state, ordinary state, and transition. Figure 2 shows an example of a STD. This STD describes a patient in a health unit, where the initial state is “waiting by attendance.” After follows the “patient care,” if the patient is healthy, the next state is “free to go,” a final state. If the patient is not healthy, the next state is “treatment,” and after this, goes back to “patient care.” This cycle repeat until the patient is healthy.

This sample of STD illustrates the capability of STD to map areal-world process flows.

A STD represents the behavior of the state machine proposed by Alan Turin (Leavitt, 2006). He proved that its state machine is a universal computer and can simulate any process symbolically described.

Besides the STD, others formal models as Business Process Modeling Notation (BPMN) (Object Management Group [OMG], 2017) can map organizational process. This work uses the STD as chosen model because its notation privileges the states and all possible transition between them, while the BPMN have the

task as your focus. Wagner et al. (2006) affirmed that a STD can map partially the dynamic behavior of a game. To treat the game rules, the software uses an inference engine (Russel & Norvig, 2013) and a KB.

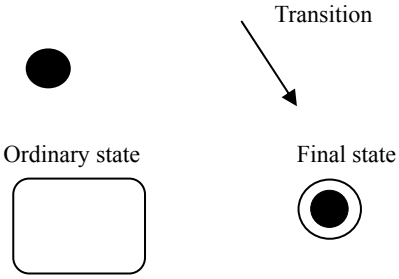

Figure 1. Components of a STD.

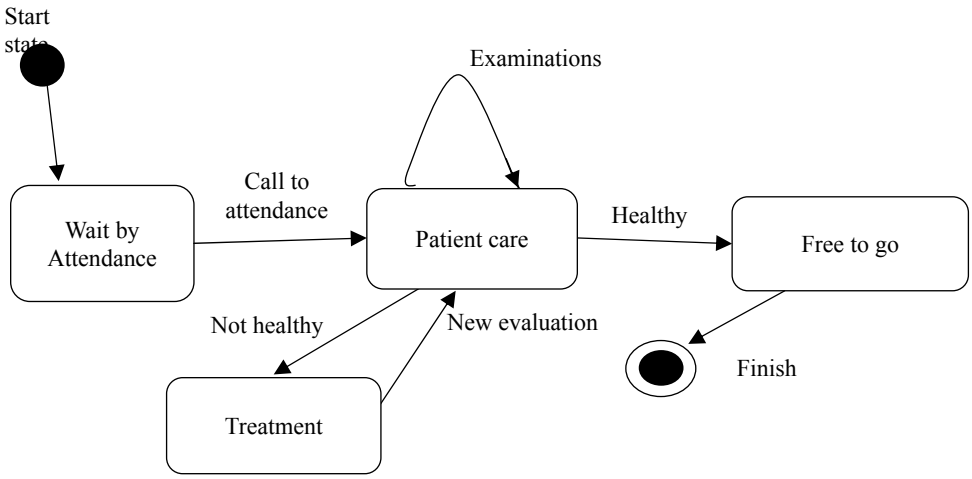

Figure 2. An example of STD.

Expert Systems

It is a kind of computer program which aims to simulate the reasoning of a specialist professional in a specific knowledge area (Russel & Norvig, 2013). An expert system has two elements: the KB and the inference engine. The first one contains rules with all knowledge of the system. The rules have the follow format *IF <condition> THEN <action>*. The inference engine can simulate the reasoning manipulating the stored rules. Figure 3 shows the mains portions of an expert system. Two kinds of users access the system, the specialist to provide the rules to the KB and the final user who provides the problems to inference machine solve.

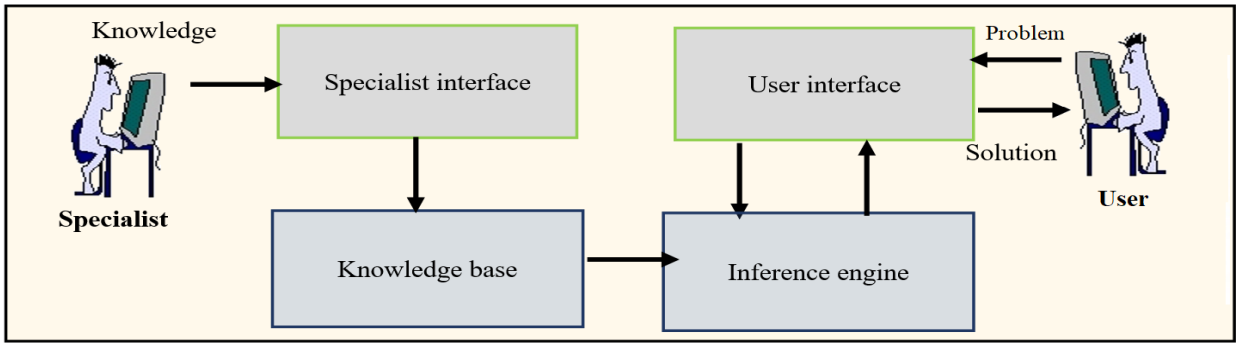

Figure 3. Basic structure of an expert system.

In this work, the intelligent module perceives the state where the game is and it analyzes the possibility to change to another state. This analyze involves the current state and the player move. If the player movement is according the game rules, the system executes the change of state.

Formal Mapping of a Game

This section presents the results of analysis performed in several didactic games. The objective is to define what are the features of a game are necessary to obtain its complete mapping. A represented game in a formal model must contain all necessary information to its simulation. This section also shows a proposal of a notation able to map a game.

A game is a set of chained states with several paths between the game initial state and its several final states. A state represents a specific situation in the game. There are a finite number of possible states in a game. The player starts in an initial state and goes by others states until reach a final state. Figure 4 shows the entity Company X with three different states: the profitability, difficulty, and bankruptcy. This entity has three variables: the sales, purchases, and expenses. The combination of variables values characterizes one of the three cited states to Company X. For example, in Figure 3, high sales, medium purchases, and low expenses indicate the occurrence of a "profitability" state.

Assuming the Company X entity is component of a specific game, where the initial state is difficulty. In this case, the player will try reaching the profitability state because it is the desired final state. Who reach this state is the winner and who reach Bankruptcy is the loser.

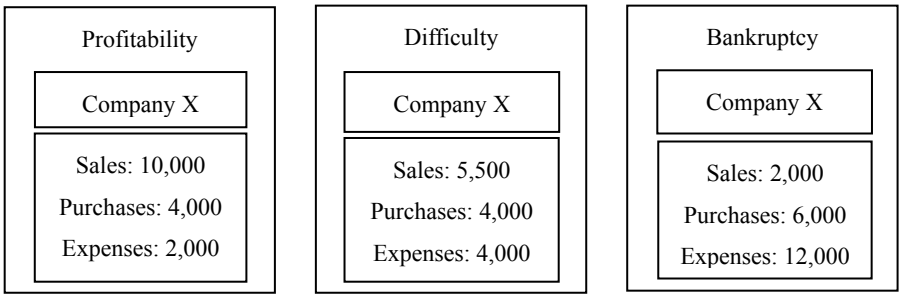

Figure 4. Representation of three possible states in a same entity.

The proposed formal model aims to map all states of the game and all possible paths between the states. The STD can map these cited elements. To complete its proposal, it is necessary a mechanism to control the states transitions. The input of this mechanism is the current state, the destiny state and the player movement. The output is the transition to next state or transition not accepted.

The game modeling process is to recognize in a specific context the important entities and, for each entity, their variables. The value of these variables defines the game states. Besides this, the process also must recognize the occurrences responsible for changing the value of the variables and, for consequence, by change the game state. These changes are the transitions. A finite set of entity, states and transition can represent any organizational process. The Turin machine (Wagner et. al., 2005) shown it is true.

Analysis of Requirements That Define a Game

Summarizing the result of analysis, following are the requirements of a game:

1. A game has a finite and small set of entities which defines its states. There are one initial state and several middle and final states;

2. An entity of a game has a set of attributes. The values of these attributes define entity states;

3. Play a game is the act of a player manipulate the game entities looking for an optimal final state;

4. The union of all entities state defines the game state;

5. There is at least one possible path between the initial game state and any other state of the game;

6. Transition is a changing from the current state to another game state. It happens when the player manipulates the value of entities attributes;

7. The games rules discipline the transitions between states. If a player move infringes a game rule, the transition cannot be performed;

8. All success player moves implicate in a transition. The player supply information to change the values from the entity attributes;

9. Besides from the player move, another kind of transition can happen in the game. Random events change the values of attributes characterizing a transition;

10. The initial values of the entities attributes in the game characterize the initial state;

11. After each game transition, the game rules permit to calculate the score of players;

12. When a player reaches a final state the game ends. The players scores calculated in this moment are the game result;

These requirements describe all observed aspect in analyzed games. Finally, it is important to observe that the number of states and of transition in the game should be small, because how greater this number is, more complex the game will be.

Formal Mapping of a Game Elements

The requirements described in previous subsection aid to create a formal notation to map a game. To facilitate the reader, each item conceptualized is underlined in the text. The proposed notation is:

1. The element game does not need to participate of the notation because the game is the own formal model;

2. Denotes entity by $\underline{E_1}$, $\underline{E_2}$, ..., $\underline{E_n}$;

3. The attributes is the element which characterizes an entity. An entity should have a finite and small number of attributes. Each attribute contain a unique informational value at a time. Its denotation is E_n: {E_n:$\underline{A_1}$, E_n:$\underline{A_2}$, ..., E_n:$\underline{A_n}$}. Where A_j is the name of attribute;

4. In the game context, there are two kinds of state founded in analysis, the individual state of an entity and the state of a game. The values of entity attributes define an individual state. The set of all individual states defines the game state. The formal model maps just the game states. Its notation is S_i, S_1, S_2, S_3, ..., S_n, S_{f1}, S_{f2}, ..., S_{fn}. Where: S_i is initial game state, S_k is middle game state, and S_{fm} is final game state;

5. The game rules work as functions applied to entities attributes. These functions modify the values of attributes and return a value true indicating permission to transition of state. The notation of a rule is $O \leftarrow R_k$ (E_m:A_n, E_{m+1}:A_{n+1}, ..., E_{m+k}:A_{n+k}) Where R_k is the *k-th* rule of the game, O responds indicating if movement obeys or not to the rule, and E_m:A_n is entity attributes manipulated by rule;

6. The model uses logical expression or procedural commands to express the behavior of the rules. This work describes how to create a rule specification in the next section;

7. The Transition is an event that causes a changing of a state to another. The transition only occurs if all rules relate to it return positive response. The notation of a Transition is T_i ($S_k \rightarrow S_j$; R_1, R_2, ..., R_n). Where S_k is state before transition, S_j is state after the transition, $R_1 \ldots R_n$ is set of rules which are related to transition;

8. These seven aspects described are enough to represent a game. This notation has all details to permit a computational simulation of the game. Finally, the expression below can resume the game representation: *Game* (*Name*; E_1, E_2, ..., E_m; S_1, S_2, ..., S_n; T_1, T_2, ..., T_n; S_i, S_{f1}, S_{f2}, ..., S_{fn}). Where *Name* is the name of the game, E_k is Set of entities that compose the game, S_k is intermediates states of the game, T_k is transition which can occur in the game, S_i is initial state of the game, and S_{fn} is final states of the game.

Figure 5 shows a graphical model able to represent the dynamic part of a game. To obtain the dynamic game meta-information the system user will draw a similar graph in the computational interface. It is observed that this graphic is a STD, which represents all game states and all possible transitions between the states.

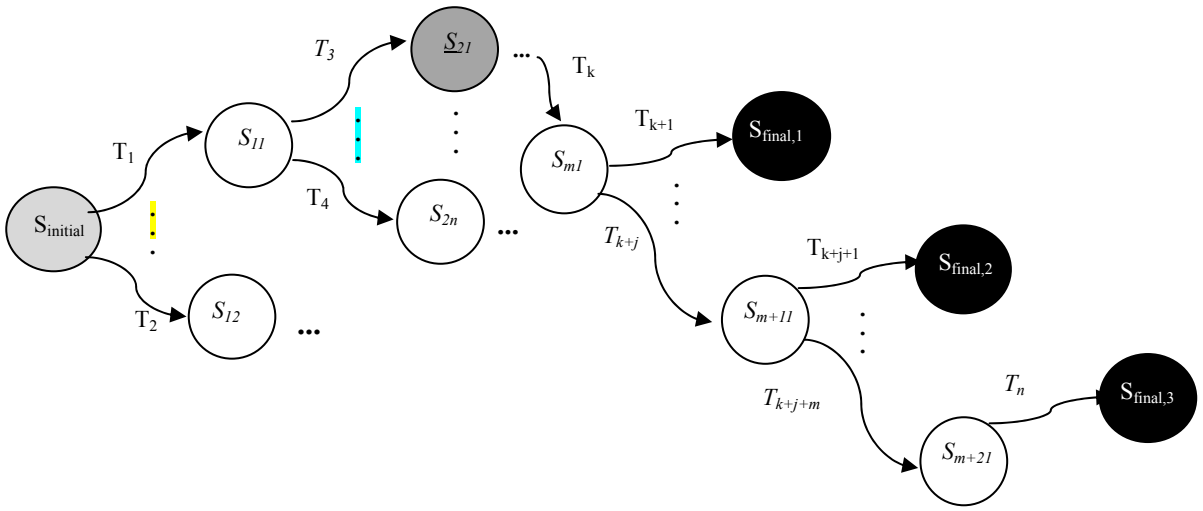

Figure 5. Graphical representation of a game.

Besides the graph, the user must define the complete description of the transitions. Then, other interfaces offer its functionality to users. The system should have interfaces to represent the follows aspects:

1. The list of game entities and, to each entity, its attributes and the initial value of each one;

2. The relation of rules. To each rule, it is necessary map its name and logical expression which defines its behavior.

The game creator must know how to write rules using the format IF … THEN. In the implementation, the user interface also permits to define the rule behavior using a procedural description.

The Computational Implementation

This work created a software able to maps and simulates a game. The first task is the capture of game meta-information, saving this information on a software repository. After the first task, the software can simulate the game. The computational resources used to implement the software were the Java language and interfaces graphical uses the Java Server Faces (JSF) and Java Server Pages (JSP). To store a mapped game in the computer, the software uses a proprietary format and text files.

Representation of States and Transition

Figure 6 shows the interface responsible to maps all states and transitions of the game and it is possible to identify all kinds of states represented in Figure 5. The white circle represents the initial state, the black circle the final states, the gray rectangles represent the intermediate states, the white rectangles the random states, and finally, the arrows represent the transitions.

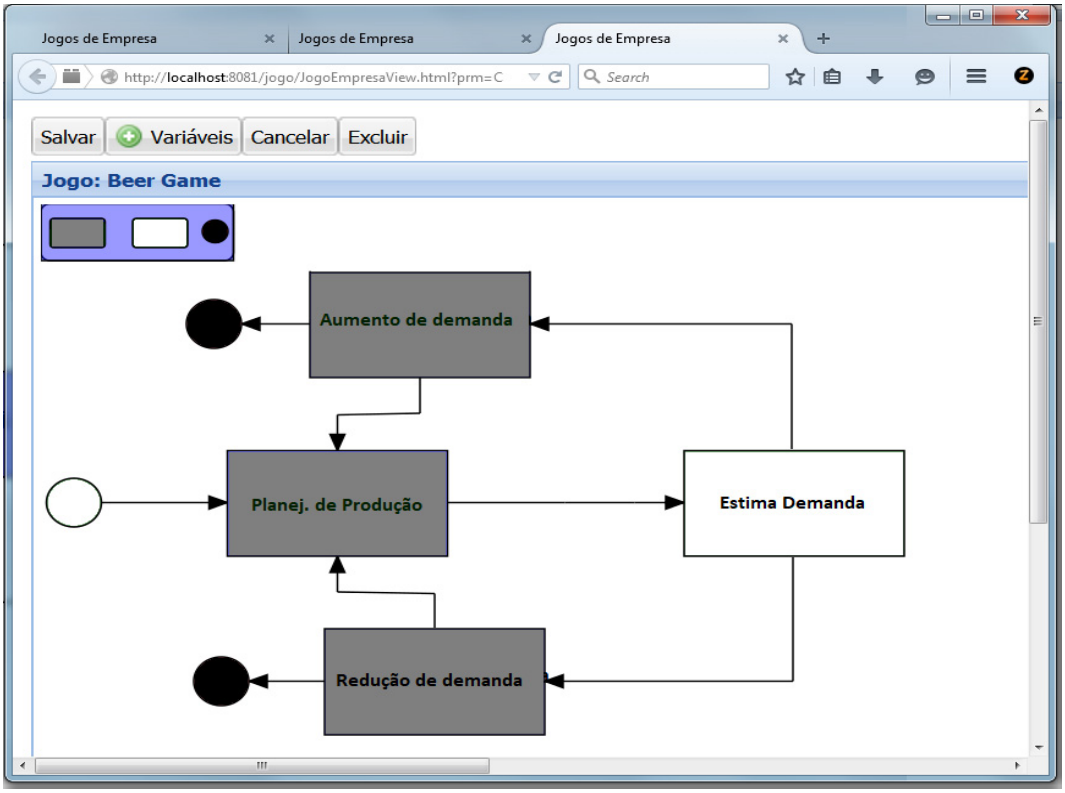

Figure 6. Interface to create the dynamic aspects of the game.

One can observe in Figure 6 that the interface represents all states and transitions of the game described in the third section of this paper. The user registers in this interface all possible moves of a player, which correspond to each arrow in the graph.

With a double-click over the gray rectangles, the user opens a new form to inform the possible game moves. This form shows all attributes of each entity registered in the model. Next subsection shows how to register an entity and its attributes. The user can mark which attributes must appear in the game form. The marked attributes can appear just to inform the game situation (marking the *Rótulo* checkbox) or to user modify its current value (marking the *Editável* checkbox). Figure 7 shows the moves interface and how the user can configure the game simulation interface.

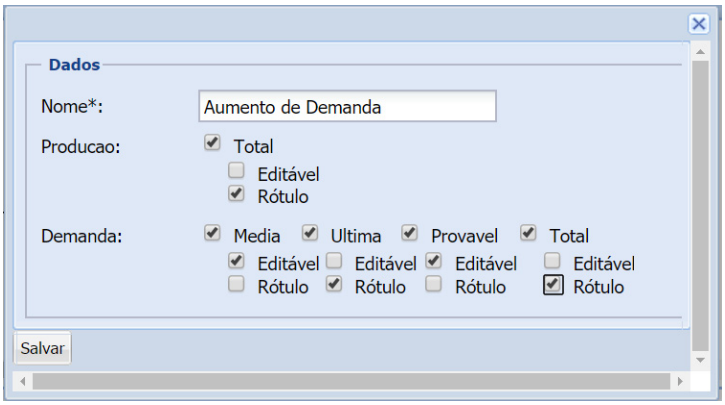

Figure 7. Define the attributes changeable by player.

Figure 7 shows that the player will can modify the attributes *Media* and *Provavel* when the game is in the "*Aumento de Demanda*" state. Other attributes marked *Ultima* and *Total* will appear, but just to illustrate details of the game context.

Besides its dynamic aspects of the game, the proposed computational interface needs to map the behavior of transitions. The player's decisions cause transitions, but these transitions must observe the game rules. Thus, aiming to map completely the game, a complementary structure is available to maps the entities, its attributes, and the games rule, because the STD does not map it.

Complementary Structure

Two different interfaces complete the mapping process of the game: the first one to register the entities and its attributes and the second one to register the game rules.

Figure 8 illustrates the first one. This interface maps all entities of the game, as well as its attributes. Figure 8 shows that in the upper half of screen, the user can register the entities, and in the low half of screen, the user registers its attributes.

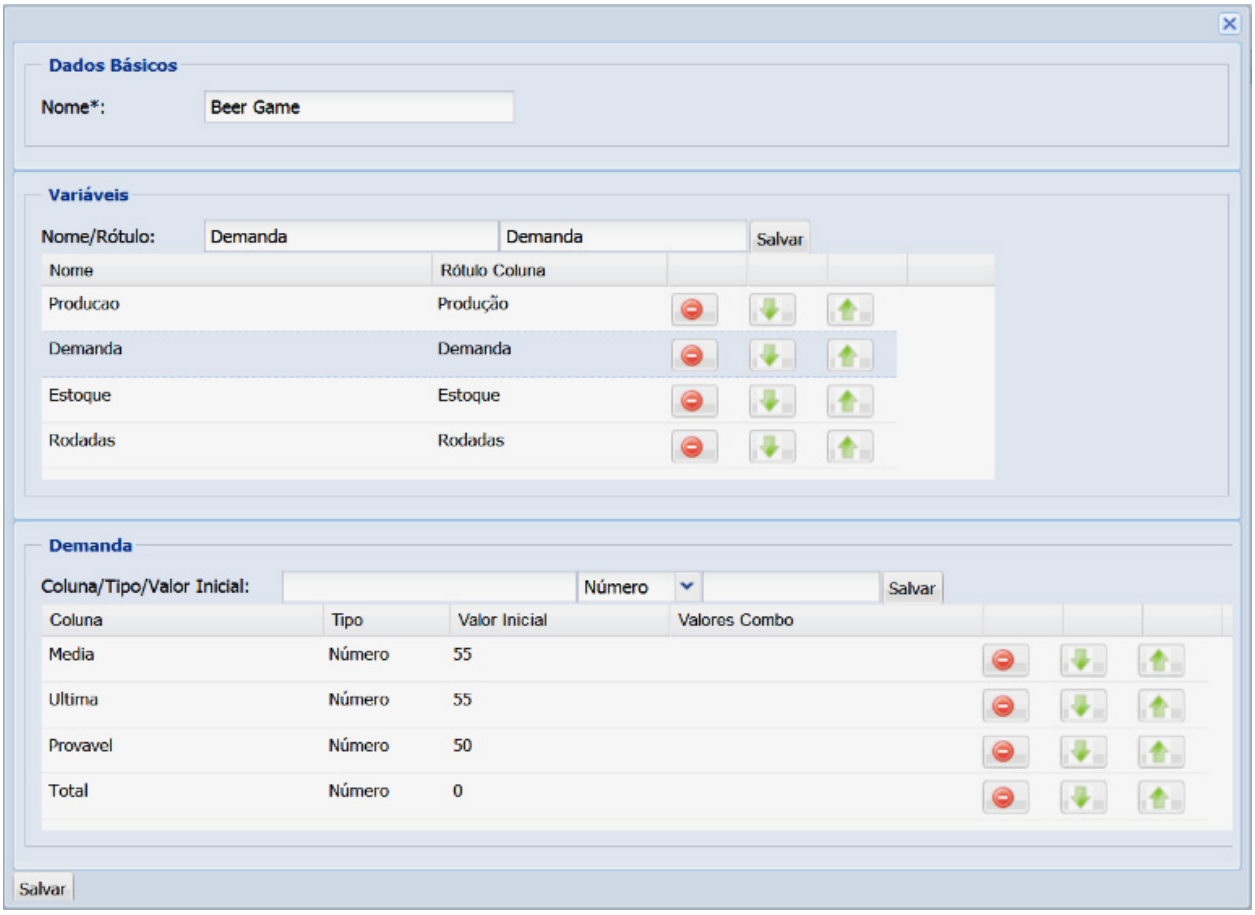

Figure 8. Interface to the registration of entities and variables of the game.

Figure 8 shows the game "Beer Game" with four entities: *Producao*, *Demanda*, *Estoque*, and *Rodadas*. The current entity appears highlighted in the interface. The current entity is *Demanda* which has four attributes, *Média*, *Último*, *Provável*, and *Total*. The user must define an initial value to each attribute registered in the software. The initial values of all attributes of the game define the initial state of the game.

The second interface registers the game rules, creating its KB. The software permits the codification of rules using the Java language or logical expression with the follow syntax: *SE <condition> ENTÃO <action>*. The software interprets and executes the Java commands, and to logical expression, it uses an inference engine to resolve the rule. This engine is like the used in expert system.

Figure 9 shows how to create the game rules. In this case, the game creator chose the Java language to create the three first rules of the game and a logical expression to create the last rule. One can observe in the codification that the procedural commands change the values of attributes causing a transition of state in the game. The last rule also changes the value of attributes, but only if the test satisfies the specified condition.

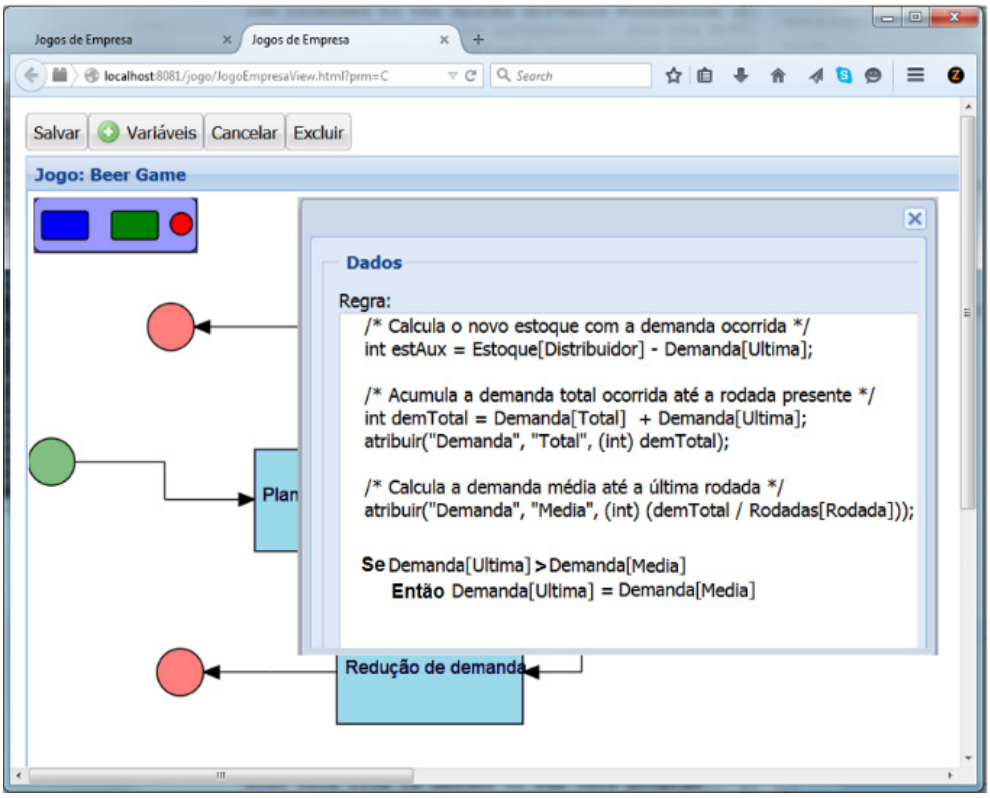

Figure 9. Knowledgebase with rules that control the game.

The Game Simulation

After the registering of the game, it is ready to be simulated. With formal model adopted in this work, the simulating process is equivalent to a function able to walk through the states, where the transitions are the path between them.

During simulation in the Web environment, the player will see the screen shown in Figure 10. In this form, the player informs its moves typing the new values to changeable attributes and clicking in the button transition. For each possible transition from a state the form will show a button corresponding to it. Besides the register of moves, this screen also shows on its top right the score of player.

The teacher can choose where it will apply the game. It can apply the game in a computer lab or in the student home by internet. This feature of the product gives it a ubiquitous character. Therefore, the proposed product is useful for classroom courses or to e-learning.

Figure 10. Game simulation interface.

Conclusions

This paper presented an approach to mapping and simulating educational games. The purpose is providing a generic environment to create games and play them on internet. The created solution used two models to map a game: the STD and a KB, where the STD models the dynamic part of the game and the KB contains all rules that control the game. These models map states and transition between states, beginning in an initial state and walk by states until reach a final state. The rules control the flow of the game, where a transition just occurs if the rules permit it. Besides the game flow control, the rules also calculate the score of the game.

The proposed model permitted the implementation of a software able to create and simulate a game. The simulating of a game consists in walk through a STD while the inference engine verifies if the flow obeys the rules of the game. Besides, the game has a user-friendly interface, where a teacher can create its own games. The registered games are available to the students on internet.

This work presented an example of how to create a game in the implemented software. Aiming to show how easy is to use software, this paper used the well-known "Beer Game." That game involves organizational processes as production planning and product selling, permitting a comprehensive test. The simulation of the game worked very well.

Finally, as the teacher creates your own games, this work concludes that the created software is a powerful resource to get better the didactic process in its courses.

References

Bernard. (2017). *Bernard simulação gerencial* (Bernard managerial simulation). Retrieved February 10, 2017, from http://www.bernard.com.br

Gramigna, M. R. M. (1993). *Jogos de empresa* (Business games). Brazil: Makron Books.

Harel, D. (1987). State charts: A visual formalism for complex systems. *Science of Computer Programming, 8*(3), 231-274.

Havey, M. (2005). *Essential business process modeling*. O'Reilly Media.

Leavitt, D. (2006). *The man who knew too much: Alan turning and the invention of computers*. London: Atlas Book.

Lopes, P. da C., & Souza, P. R. B. (2004). *Jogos de negócios comoferramenta para construção de competências essenciais às organizações* (Business games as tools to create essentials competencies to companies). Retrieved from http://www.ead.fea.usp.br/Semead/7semead/paginas/artigosrecebidos/Ensino/ENS18B-Jogos_de_Negócios_como_ferramentas.PDF

Object Management Group (OMG). (2017). *Object management group*. Retrieved July 31, 2017, from http://www.omg.org

OGG Simulação Empresarial. (2016). *Simulador para pequenas empresas* (Simulator to small companies). Retrieved January 30, 2016, from http://ogg.com.br

Russell, S., & Norvig, P. (2013). *Inteligência artificial* (3rd ed.). Campus, Brazil.

Sauaia, A. C. A. (1995). *Satisfação e aprendizagem em jogos de empresas: Contribuições para educação gerencial* (Satisfaction and learning in business games: Contributions to managerial education) (Doctorate thesis, Universidade de São Paulo [USP], São Paulo).

Simulare. (2017). *Simulação de gestão de negócios* (Simulator of business management). Retrieved June 12, 2017, from http://simulare.com.br/page/

Sommerville, I. (2003). *Engenharia de software* (6th ed.). São Paulo, Brazil: Addison Wesley.

Vicente, P. (2000). *Jogos de empresas: A fronteira do conhecimento em administração de negócios* (Business games: The frontier of knowledge in business administration). Brazil: Makron Books.

Wagner, F., Schmuki, R., Wagner, T., & Wolstenholme, P. (2006). *Modeling software with finite state machines: A practical approach*. Florida, USA: Auerbach Publications.