Software Agents in Educational Board Games

BOYAN BONTCHEV\textsuperscript{1}, SERGEY VARBANOV\textsuperscript{2}, DESSISLAVA VASSILEVA\textsuperscript{1}
\textsuperscript{1} Department of Software Engineering, Sofia University, Faculty of Mathematics and Informatics, 5, J. Bourchier blv., 1164 Sofia, BULGARIA
\textsuperscript{2} Institute of Mathematics and Informatics, BAS, 1113 Sofia, BULGARIA
bbontchev@fmi.uni-sofia.bg, varbanov@fmi.uni-sofia.bg, ddessy@fmi.uni-sofia.bg

Abstract: In last decade, educational games such as quizzes, puzzles and quests have provoked an increasing interest in new methods of learning and appeared to be an appeal motivating students to study in a way rather different than the traditional one. Board games manipulate figures a surface according predefined rules in a competitive play mode with race conditions. Traditional quizzes, puzzles and quests can be presented as multimedia board games, often with artificial agents simulating the opponent or a collaborator. Authors present a formal model of board games serving for educational purposes, and show the place of artificial agents within the game construction and possible ways of agent’s realization.

Key-Words: Agents, quiz, board game, e-learning, intelligent

1 Introduction
Recent developments in technology-enhanced learning environments and widespread interest in distance learning are forcing educators to consider new approaches to teaching and learning. Computer-based learning environments are beginning to replace, or at least supplement, traditional face-to-face instruction. A stable trend in e-learning evolution is usage of different approaches complement to the traditional instructional learning design with educational games [1]. When designed carefully and appropriately for the audience, games can be effective at motivating learners and maintaining their attention. Educational games can be entertaining and, at the same time, cognitive, as their goals, constraints, rules for payoff, and consequences are precisely defined [2], [3]. The main objective in designing an educational game is to encourage learners to improve their knowledge by appealing them to solve various problems of particular domain area or practice, for improving their creative thinking [4]. Mark Prensky describes in [3] how games give pleasure and emotions during the play process, strengthen creativity by imposing conflicts and competitions, use rules and set goals for provisioning structure and motivations and, thus, satisfy the ego of the players/learners in an interactive way.

Usually, nowadays educational games are represented by quizzes, puzzles, quests and problem solving staging [5], [6]. Board games use surface (board or map) for moving figures according predefined rules using counters or dices and appear as type suitable for educational purposes [7]. Board games may be played by a single player alone or against an intelligent, software agent as real opponent [7]. Several players may learn interactively while playing a game for fun.

The paper aims to describe both principal model and realization (by means of software agents) of problem-oriented board games, representing quizzes, puzzles and quests. It explains in brief a new, formalized model of such types of games helping analysis and design of such games and their software implementation. The model is planned to be used as a basic paradigm for the development of a multimedia game platform for an e-learning system providing adaptive courseware. Special attention is put on usage of intelligent agents within the platform for simulating the opponent of the player. There is described the practical implementation of the agents by suitable software tools and platforms.

2 Motivational Background
There are many good examples of well designed educational games used to support e-learning processes. The majority of them are quizzes, puzzles and quests, because they suppose a simple construction process and are very useful for self-assessment and official test.

2.1 Quizzes as Educational Games
Quizzes are widely used either to be played as games for fun or to be solved within the e-learning knowledge assessment process in two main ways: to make self-assessment tests or to do final exams. There are many examples of commercial and free tools for educational quiz design and run-time quiz administration such as Quiz Center, Quiz Builder, Wondershare QuizCreator,
and Tanida Quiz Builder [7]. Some of them provide means for automatic generation of questions and, as well, for automatic assessment of answers [8]. On the other hand, many quiz builders provide authoring tools for creation of quizzes [9], [10]. There are some authoring tools offering opportunities for creation of parameterized questions [11]. Finally, there are some combinations between board games and quizzes as applying board rules for navigation within a quiz. Navigation rules may control movements from one question to another, or manage choosing a question, e.g. choice of complexity [12].

In last decade, with new achievement in the area of adaptive e-learning platforms there were developed some adaptive quizzes and educational board games. A good example of such a game in the area of adaptive learning is ELG [9]. ELG allows presentation of different learning activities in the form of board games. Learners are supposed to improve their performance and broaden their knowledge simply by participating in such games. The ELG provides an authoring tool for personalization of each game according various issues such as learner’s level of knowledge, preferences and educational goals. The adaptation is based on rules and depends on shown knowledge and defined goals. It is used to manage transitions from one game level to another. Other systems for adaptive quiz control are QuizGuide and QuizJet [13] which have been developed as adaptive systems. They help students to select self-assessment quizzes most relevant to individual needs and learner’s preferences. These systems are able to choose the most important training topics by mean of adaptive navigation, which does help weaker students to improve their performance. Another class of games called active learning games is based on a pedagogical approach of actively involving students into key concepts of a problem. The Genetic Algorithm Game [14] is an example of such an active learning game. It is applied in engineering education with very positive results in student experience and conceptual learning.

Newly proposed board (or map) games use rule-driven approaches for both presentation and control. Such approaches rely on profound knowledge of issues of course material for achieving safe navigation through a map or board [9]. The concepts of these types of games ensures many benefits such as taking turns by rolling dices, varying questions in terms of difficulty, and applying various strategies for selection of a question by individual player [12].

2.2 The Role of Software Agents in Educational Games

Educational games can be played in various ways - by multiple users together, by a single user alone as a normal quiz and even by a single user against a simulated player - by means of an intelligent, software agent. Software agents are already proven as an architectural approach for building distributed, flexible (sometimes open) software applications. This approach definitely has its place in the area of serious games as far as it fits quite naturally to human’s way of thinking about virtual world and different stakeholders. As authors have already noted in [15], intelligent agents can be used in games taking the following roles:

- Simulation engine modules - at least some of the autonomous processes of the simulation engine could be designed and implemented as software agents, they act on behalf of the game. They are part of the simulation machine and typically they have relatively constant behavior and are included in every game run.
- Competitive non-human players - these agents take the role of players, competing with the human players. Typically, their number, type and behaviour can vary from game run to game run. The game moderator instantiates and sets these agents to achieve specific indirect influence onto the human players. Thus it can be said that they act on behalf of the game moderator.
- Collaborative non-human players - in long time running games, such agents can be used to do routine tasks on behalf of the human player.
- Human player’s advisors - agents providing human players with hints, advice and additional information when needed or requested. They act on behalf of game’s mentor.

Therefore, there is diversity of agent’s roles observed. Let us note that agent’s complexity could also vary greatly. For example, in a board game with limited state space, the agent might exploit well known state space search procedures, e.g. min/max or alpha/beta procedures for opponent’s role. In this case the agent would be relatively simple. And at the other side, in a game with high complexity and very large and multidimensional state space, the agent probably will have to deal with a virtual world model; with knowledge about the game; with adaptation and learning; even with a model of itself if it acts in a community of social agents.

Thus, the software complexity and accordingly technologies and tools needed for agents’ realization would also vary. To apply different technologies for every single implementation is not effective. We need a simple approach with great potential, possibly an open, distributed architecture. Recent advances of software technologies, stimulated by Web services and semantic Web areas provide approaches, tools and ideas from which we could benefit.
3 Educational Board Games

Quizzes, puzzles and quests can be modeled as special board mini-games, where the board may be any map with positions and figures (objects) over it, moved by rules for manipulation and causing resulted effects [12]. A runtime execution of such a quiz board game model has been proposed in [7]. The model sketched in next section is general enough in order to allow description and execution control of logical problems to be solved and, in general, of any learning activities and their workflow.

3.1 A Formal Educational Board Game Model

In any board game, objects are figures of different types (i.e. classes) and may be allocated on some of the positions over or outside of a map. The number of instances (power) of each object type (class) may be one or more. The gamer is able to do some actions over the objects such as filling text at some input fields, single or double mouse click over a figure or position, drag and drop an object from source to destination position, or something more complex as rotation of a figure [16]. Each action may be executed only under some predefined rules (conditions) for the given object type. These preconditions may concern the manipulated object itself (whether objects of its type could be moved, clicked, rotated, etc.) and/or the destination position and positions between the initial and destination one (for move, i.e. drag-and-drop actions) – for example, capacity checks of destination position. Finally, after each allowed action, one or more resulted events may be fired depending on the action success status. Together with these result events, the game execution engine checks whether the final objective of the game is achieved. This check is done at the end of each action in order to verify the finish conditions, as one of the resulted effects for the action. Thus, the model defines a quiz board game as [7]:

\[
\text{Game} = \{\text{GB, PT, P, OT, O, InitDisp, A, R, E, FC, Res}\},
\]

where

- GB is a contextual game background image (visible or not);
- PT = PT₁, PT₂, ..., PTₖ - a set of all possible types of positions; the number of instances (i.e. of positions) of given type PTₙ may be 1 or a integer number;
- P = P₁, P₂, ..., Pₙ - a set of positions (of equal or different types, k≤n) with predefined properties which may be checked by a checkProperty(P_i) and set by a setProperty(O_j) functions, where properties are type, capacity, title, content, visible, selected and location;
- OT = OT₁, OT₂, ..., OTₖ - a set of all possible types of arbitrary objects (figures); the number of instances (i.e. of objects) of given type OTₙ may be 1 or a integer number;
- O = O₁, O₂, ..., Oₘ - a set of object (figure) of any type of the set OT with predefined properties which may be checked by a checkProperty(O_j) and set by a setProperty(O_j) functions, where properties are type, title and location;
- InitDisp – a two dimensional matrix of initial disposition of each object Oᵢ ∈ O over one position Pⱼ ∈ P;
- T = T₁, T₂, ..., Tᵢ, ..., Tₘax – a set of time discrete, or steps, when the moves are played; may be restricted or not;
- F = F₁, F₂, ..., Fᵢ, ..., Fᵣ – a set of functions over P and O, such as containsTyped(Pᵢ, OTₙ, Tᵢ), containsAny(Pᵢ, Tᵢ), get(Pᵢ, Oᵢ, Tᵢ) and put(Pᵢ, Oᵢ, Tᵢ);
- A = (A₁, A₂, ..., Aᵢ, ..., Aₙ) – a set of player’s actions, such as mouse click, double mouse click, and move (drag-and-drop);
- R = (R₁, R₂, ..., Rᵢ, ..., Rᵣ) - a set of rules each under specific conditions over F, for object type manipulation at specific position for given action; rules may differ from position to position and from action to action;
- E = (E₁, E₂, ..., Eᵢ, ..., Eᵣ) – a set of effects to be applied after each action over object of specific type residing given position;
- FC – final condition – for example, arithmetic expression over F and attributes of P and O, check for the game result, etc. If the final condition returns true, the game is over;
- Res – result as an arithmetic condition over F and attributes of P and O.

3.2 A Sample Quiz modeled as Board Game

In [12] there is shown that quiz questions, puzzles and quests may be developed as board mini-games. Questions with one exact answer may be presented by a single position with title equal to the question and empty content editable through mouse click. Question with answers of type one of many, many of many, and filling blank have a straightforward and easy presentation. For such types of questions, answers are going to be represented as textual or multimedia objects and the player have either to select them by clicking or to move one or many of them to the position of right answers. Presentation of question of type “order of answers” and matching problems supposes more interesting presentation. Fig. 1.a represents the start view of a quiz
board game of type “concept matching” similar to puzzle games. Here, players have to discover the right matches between dark polygons over and white stars below and, next, to move each of the stars over the right polygon. The three matches are of type 1 to 1 mapping between positions (polygons) and objects (stars). Fig. 1.b depicts a successful game finish. Both wrong and successful finishes may be accompanied by multimedia effects as well as each right/wrong placement of a star over a polygon. There is no need of time steps definition. Thus, the model is as follows:

- **PT** = Pentagon, Hexagon, Octagon, ServicePosition (defined by the author);
- **P** = P1, P2, P3, P4, P5, P6; setType(P1, Pentagon), setType(P2, Hexagon), setType(P3, Octagon), set P4, P5, P6 as ServicePosition’s and invisible (used only for service purposes); all the capacities are equal to one.
- **OT** = 5star, 6star, 8star (defined by the author);
- **O** = O1, O2, O3, setType(O1, 5star), setType(O2, 6star), setType(O3, 8star);
- **InitDisp** = O1 in P6, O2 in P4, and O3 in P5;
- **R** = Rj(move, Pj, Pi, Oj), for i=4..6, ii=1..3, j=1..3, apply E1 and E2;
- **E** = E1=get(Pi, Oj, 1), E2=put(Pii, Oj, 1);
- **FC** = {containsAny(P4) = containsAny(P5)=containsAny(P6) = empty};
- **Res** = {if (containsTyped(P4, O1)==1 and containsTyped(P5, O2)==1 and containsTyped(P6, O3)==1) then result=true else result=false} – fig. 1.b.

**4 The Software Agent Architecture**

In previous work [15], we have used a three-layered architecture of the software agent shown in fig. 2. The first layer contains methods for interfacing the virtual world in which the agent acts. The second layer represents the model of the virtual world, which the agent builds and uses. The third layer contains the agent’s decision-making functionality, which applied to the virtual world model implements the decision-making process. For the implementation of those three layers, different technologies were used, accordingly:

- Java methods to implement agent’s sensors and effectors, communicating with the game server via CORBA;
- Protégé [17], as knowledge base management system to implement the virtual world model;
- Algernon [18], as forward-backward rule based inference engine to implement the agent’s decision-making functionality.

That approach, although fruitful, left the impression that in many cases the full power of Java, Protégé and Algernon would not be needed. Therefore, this three-layered architecture could be implemented by more “lightweight” means.

**Fig. 2: Three layer agent’s architecture used in PRIME project [15].**

Recent advances of software technologies, stimulated by Web Services and Semantic Web areas, provide interesting approaches, tools and ideas from which we could benefit in this aspect. The ideal solution would consist of maximum lightweight tools for the trivial cases, and enough openness to provide for easy inclusion of heavyweight tools as Protégé and Algernon if needed. Currently, our choice falls on Persevere Server [19] - an object storage engine and application server (running on Java/Rhino) that provides persistent data storage of dynamic JSON data in an interactive server side JavaScript environment (fig. 3). Among its many features, the following have to be noted:

- Create, read, update, and delete access to persistent data through a standard JSON HTTP/REST Web interface;
Comprehensive referencing capabilities using JSON referencing, including circular, multiple, lazy, non-lazy, cross-data source, and cross-site referencing for a wide variety of object structures;

- Data integrity and validation through JSON Schema;
- Class-based data hierarchy - typed objects can have methods, inheritance, class-based querying;
- Full access and control over Java libraries via JavaScript.

5 Conclusions and Future Works

The presented model of quiz questions, mazes, quest and other logical problems as a multimedia board game offers a plenty of benefits. It is very general and allows presentation of more complex logical problems which have to be solved by actions delivered by the player according execution of complex rules under context conditions and functions [12]. Note, that the types of positions and objects are arbitrary, thus, there are no restrictions about them. Moreover, it enables description as a board game of a complex set of e-learning instructions and activities within given domain context. Examples for such scenarios are activities typical for visiting virtual world, following a cooking receipt, locating an object over geographical map, etc. Therefore, the game model should be proven by building both a game design tool and execution engine which runs the game model executing rules for each player’s action under certain conditions.

The aim of the formal game model is to serve as a base for construction of both game design tool and runtime control engine for educational game execution. The formal description will be used for creations of definitions of board games and, next, for execution of the formal rules by the engine. The presented approach for realization of software agents for educational board games combines ideas from open architectures, distributed and agent programming, and Web services. In our case, usage of such software application would shorten the design and implementation phase and would provide smooth implementation iterations at later stages. For example, a simulation engine module providing market price of some product could be implemented as random-walk generator at first. Later this module could be evolved in more complex model establishing relation to supply and demand dynamics. Similarly, an intelligent software agent could be primitive in the beginning and later be replaced with a full-blown artificial intelligence agent, if appropriate.

The proposed combination of technologies and tools provides also good ground for applying extreme programming techniques. It facilitates usage of agile principles and spiral approaches to project management. The last is very important, because the model is going to
be used as a basic paradigm for development of a multimedia game platform for an e-learning system providing adaptive courseware. The model should be extended by metadata for each type of game in order to describe what type of user a particular game is appropriate for. Thus, different types of learner will be offered different serious educational games [20] according to their psychosocial profile, interests and knowledge.

Acknowledgments
The work reported in this paper is supported by the ADOPTA project funded by the Bulgarian National Science Fund under agreement no. D002/155, and partially by the SISTER project funded by the European Commission in FP7-SP4 Capacities via agreement no.: 205030.

References: