International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:6, 2010

Educational Quiz Board Games for Adaptive E-Learning

Boyan Bontchev, Dessislava Vassileva

***Abstract*—**Internet computer games turn to be more and more attractive within the context of technology enhanced learning. Educational games as quizzes and quests have gained significant success in appealing and motivating learners to study in a different way and provoke steadily increasing interest in new methods of application. Board games are specific group of games where figures are manipulated in competitive play mode with race conditions on a surface according predefined rules. The article represents a new, formalized model of traditional quizzes, puzzles and quests shown as multimedia board games which facilitates the construction process of such games. Authors provide different examples of quizzes and their models in order to demonstrate the model is quite general and does support not only quizzes, mazes and quests but also any set of teaching activities. The execution process of such models is explained and, as well, how they can be useful for creation and delivery of adaptive e-learning courseware.

***Keywords*—**Quiz, board game, e-learning, adaptive.

I. Introduction

NOWADAYS, development of Internet technologies in the area of e-learning allow in some cases traditional training to be replaced or supplemented by the online one. E-learning enables in educational process to be included materials with multimedia content such as simulations, games, movies and more. In this article, authors consider usage of games in the e-learning process. Games are a very effective mean to attract attention and retain interest and can be simultaneously entertaining, educative and cognitive. Therefore one of the new trends in e-learning is the development of different approaches complement the traditional instructional learning design with educational games [1]. These educational games can be seen as a mean for determining several of learning activities, for one or more players, with predictable outcome, where goals, constraints, rules for payoff, and consequences are precisely defined [2, 3]. They can be used successfully in school education as well as in universities and professional training courses. The main objective in designing an educational game is to encourage students to improve their knowledge by making them to solve various problems of particular scientific area or engineering practices and, thus, by improving their creative thinking [4]. Mark Prensky [3] indicates some of their most important characteristics:

- Games give pleasure and emotions during the play process
- Games strengthen creativity by imposing conflicts, competitions and challenging problem solving
- Games use rules and set goals for provisioning structure and motivations
- Games satisfy the ego of the players/learners by appealing win states
- Games are interactive and may be adaptive which engages the learner

Some of the main types of educational games developed so far are quizzes, puzzles and problem solving staging [5, 6]. Authors propose board games to be also included, which supposes moving figures across a surface (board or a map) using counters or dices. Board games may be played by single users alone or against an artificial agent presenting a real opponent [7]. Most often they are played by two or more players thus enabling participants to learn interactively from one another while playing for fun. There are designed educational games, which are combination between board game and quiz as they apply board rules for navigation within a quiz (to move from one question to another) and for control of the questions (choice of complexity) [8, 9].

Our paper aims to describe management process of more general, multimedia-rich and problem-oriented quiz games, which are created by means of board game instruments. Moreover, it explaines a new, formalized model of such types of games. This model will assist creation of a clearer specification for software implementation and can be used for more precise analyze of performance of these type of games compared with other ones. It supports creation, management of internal logic and interpretation of results of quizzes, puzzles, quests, logical problems as well as instructional set of teaching activities presented as a race game. For game control, an adaptive strategy is used based on result of dices for random selection of complexity of the next question. The model is going to be used as a basic paradigm for the development of a multimedia game for an e-learning system providing adaptive courseware. Therefore, our model should include metadata for each game. The main purpose of these metadata is to describe what type of user is appopriate for a particular game. Thus, different types of learner will be offered different games according to their psychosocial profile and interests.

II. Related Works

Quizzes are a great way to play games for fun, to make self-assessment tests or final exams. Therefore they are widely used in e-learning knowledge assesment process. There are a

B. B. Author is with the Department of Software Engineering at Faculty of Mathematics and Informatics, Sofia University St. Kl. Ohridski (e-mail: bbontchev@fmi.uni-sofia.bg).

D. V. Author is with the NIS at Sofia University St. Kl. Ohridski (e-mail: ddessy@fmi.uni-sofia.bg).

number of commercial tools for quiz designe such as:

- Quiz Center[1] – a free tool mainly for creating and administering of online quizes.
- Wondershare QuizCreator[2] – allows to combine test quiz with multimedia and delivers a math equation editor. Moreover it supports SCORM standard
- Tanida Quiz Builder 2[3] – creates Flash based educatinal quizes, which can be used as online or desktop applications
- Quiz Builder[4] – similar to Tanida Quiz Builder; can create Flash based quizes and provides automatic report of the test result

There are several frameworks and tools to be used for design of educational quizzes. Some of them allow automatic generation of questions and, as well, automatic assessment of answers [10] while others propose creation of quizzes by authoring tools [9, 11]. Other authoring tools offer opportunities to create parameterized questions [12].

In recent years, with the enforce of the paradigm of adaptive e-learning there were developed several adaptive quizzes and educational board games. One such game in the area of adaptive learning is ELG [9]. In ELG different learning activities are presented in form of board games. Learners by participating in them improve their performance and broaden their knowledge. ELG provides an authoring tool for personalization of each game depending learner's level of knowledge, preferences and educational goals. It uses rules which, depending on shown knowledge and defined goals, manage the transition from one game level to another. Other systems for adaptive quiz control such as QuizGuide and QuizJet [12] have been developed as adaptive systems helping students to select self-assessment quizzes most relevant to individual needs and preferences of each learner. In order to choose the most important training topics, these systems use adaptive navigation, which helps weaker students to improve their performance.

Another class of games is based on the pedagogical approach in which students are actively involved with key concepts of a problem. Games of this type are called active learning games. Example of implementation of an active learning game is Genetic Algorithm Game [13]. This game is used in the area of engineering education and provides very positive results in student experience and conceptual learning.

Newly proposed map (or board) rule-driven approaches are a modern trend in quiz presentation and control. In these approaches profound knowledge of the teached course material assures safe navigation throught a map (board) [8, 9]. The concepts of these type of games ensure some benefits such as players make take turns by rolling dices, questions may vary in terms of difficulty, and there could be applied varios strategies for selection of question card for each player. Such games can be played by multiple users together , by a single user alone as a normal quiz and even by a single user against a simulated player (by an artificial agent).

[1] http://school.discoveryeducation.com/quizcenter/
[2] http://www.sameshow.com/quiz-creator.html
[3] http://tanida-quiz-builder.smartcode.com
[4] http://www.quiz-builder.com/

III. Modeling Quizzes as Multimedia Board Games

It is obvious that modern e-learning needs to involve educational games as an important element for attracting interest of learners and, as well, for developing specific skills and knowledge. Adaptive e-learning approaches make use of such games in a way adaptable to specific character of individuals or group of learners. Therefore, specification of metadata for each game appears to be here of crucial importance – if the learner's character is known, it would be possible to make choice of the mostly appropriate game for this character.

Quizzes, mazes and solution of specific problems are among most popular educational games, while board games are preferred most often for entertainment and strengthening intellectual potential. Usually, quizzes are utilized as tests for self- and/or official assessment, within the scope of given course or a specific topic. Usage of quizzes continues being within the frame of traditional quiz format, presentation and run time control. Unlike quizzes, a board game is created by special game design tools and is based on a specific strategy for problem resolution in the scope of given configuration, usually, under given race conditions. Both quizzes and board games may played by a single user and/or multiple users. More often, run time is divided into steps (passes, paces) which cannot exceed predefined time duration.

In this section, it is explained how a quiz can be modeled as a special board mini-game, with board of any form, positions, figures (objects), rules for manipulation and resulted effects. Next, it is proposed a run time execution of such a quiz board game model. The model itself 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.

A. The Quiz Board Game Model

Usually, a quiz is composed by separate questions presented at run time in given time order. Statically built quizzes have a predefined flow of quiz questions, while dynamic quizzes select questions on run time in a random way or using given criteria. Within the quiz flow, each one of the questions represents a specific problem and requires one or several answers from a predefined answer set or needs to be answered by a predefined word or sentence (filling blank question type). There are other, more interesting types of questions, requiring sorting a list of words or solving a proper mapping between objects. Finally, there are question of different type stating a general problem to be solved. Questions of the last type operate with graphical abstractions (figures) and the person solving such a quiz has to move, reorder or rotate them in order to solve the problem.

The authors aim in creation of a general quiz model, which could present all the quiz questions and processes controlling their answering. More, such a presentation should be

attractive, with high level of dynamism and appealing multimedia. On other side, the model should be general enough to allow description of any situational problem and its solution suitable for e-learning purposes. A wide range of e-learning activities should have possible and not very complex model representation such as visiting a virtual museum and shooting pictures of some pieces of arts [14], collecting or selecting objects of given type residing specific locations, discovering a specific object on a map, finding (shortest) way within a labyrinth, etc. The execution of such game models should be straightforward and not very complex, following predefined game control rules.

Generally, a quiz can be viewed on a non-restricted board, presented as an image map (contextual background specifying what should be done by the gamers) with given configuration of positions of various types over it. Normally, the capacity of each position is 1 but there are games with greater and even different capacities of positions. For grid games, configuration is defined by functions of neighborhood for all the positions. In non-grid board games, positions are spread over the background, however, even in such a case they may be viewed as aligned to an invisible grid as in graphical editors. Thus, it is possible to model dislocation of positions via grid coordinates which gives us the flexibility to move them to another grid cell when needed.

In board games, objects (figures) of different types (classes) may be allocated on some of the positions (or out of the map, on some reserved positions). The multiplicity (power) of each object type (let say – class) may be one (singleton) or greater than one. The gamer is supposed to do some actions over the objects such as filling text at some input fields, dragging and dropping an object from source to destination position, single or double mouse clicking over a figure or position, or even rotating a figure [15]. For a given object, each action may be executed only under some pre-defined rules (conditions) for the given object type. Such 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). Such conditions may check whether the destination positions and positions between it and the source position are free, or other facts. Finally, after each allowed action, one or more resulted events may be fired depending on the action success. Together with these result events, the game execution engine should check if the final objective of the game is achieved. This check should be done at the end of each action in order to verify the finish conditions, as one of the resulted effects for the action. Thus, = a quiz board game is defined as the following:

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

In this model, the building elements are as explained below.

- GB – contextual game background image, may be visible or not
- $PT = PT_1, PT_2, \ldots, PT_k$ - a set of all possible types of positions; the number of instances (i.e. of positions) of given type PT_p may be 1 or a integer number.
- $P = P_1, P_2, \ldots, P_n$ - 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(P_i) functions:
  - type – $\forall$ checkType(P_i) $\in$ PT; position type is almost always fixed
  - capacity – integer value (usually, checkCapacity(P_i)=1 and is constant), determines the maximum number of object instances which can be allocated on P_i
  - title – a string value, empty by default, may be changed by the player (for editable titles)
  - content – a string value, empty by default, may be changed by the player (for editable contents)
  - visible – a Boolean value, true by default, may be changed by the player
  - focused - a Boolean value, false by default, may be changed by the player by mouse click
  - location – a 2-dimencional location of the position on the grid (the function checkLocation(P_i)=(x, y), where x, y are integers, and usually are constant)
- $OT = OT_1, OT_2, \ldots, OT_l$ - a set of all possible types of objects; the number of instances (i.e. of objects) of given type OT_p may be 1 or a integer number.
- $O = O_1, O_2, \ldots, O_m$ - a set of objects (figures) of any type of the set OT with predefined properties which may be checked by a checkProperty(O_i) and set by a setProperty(O_i) functions:
  - type – $\forall$ checkType(O_i) $\in$ OT; object type is usually fixed but in some games (like chess) may be changed once (in chess, when a pawn reaches the final position) or more times under special circumstances; usually, there is one object for each object type: $\forall$ OT_p, $\exists!$ O_q.
  - title, content, visible, focused – like for the positions
  - location – a 2-dimencional location of the position on the grid; checkLocation(O_i)= checkLocation(P_j)=(x, y), where x, y are integers, and may change with player's moves

TABLE I
SAMPLE INITIAL DISPOSITION OF OBJECTS OVER POSITIONS

Objects / Positions	O_1	O_2	…	O_i	…	O_m
P_1	1	0		3		1
P_2	0	0		0		0
…						
P_i	3	0		2		4
…				0		
P_n	0	1		1		0

- InitDisp – a two dimensional matrix of initial disposition of each object $O_i \in O$ over one (only one!) position $P_j \in P$. Table 1 presents an example of such an initial disposition where there are more than one objects in some of the positions (though for most of the game the maximum number of objects per position is one.

For every InitDisp, it is preserved that sum of all objects in given position is less or equal its capacity.

- $T = T_1, T_2, \ldots, T_t, ..., T_{max}$ – a set of time discretes, or steps, when the moves are played; may be restricted or not; for each T_t there may be defined a maximum timeout or a delay
- $F = F_1, F_2, \ldots, F_c, ..., F_f$ – a set of functions over P and O, such as:
  - containsTyped(P_i, OT_k, T_t) – returns the number of objects of given type $OT_k \in OT$ at given step T_t (0 or an integer); contains(P_i, OT_k, T_t) <= checkCapacity(P_i);
  - containsAny(P_i, T_t) – returns an array with numbers of objects of any possible type $OT_k \in OT$ at given step T_t (0 or an integer); contains(P_i, T_t) <= checkCapacity(P_i). For example, for the initial disposition given in table 1, we have contains(P_i, T_0) = {3, 0, …, 2, …, 4};
  - get(P_i, O_j, T_t) – removes object O_j from position P_i at time T_t;
  - put(P_i, O_j, T_t) – place object O_j at position P_i at time T_t;
- $A = (A_1, A_2, \ldots, A_s, ..., A_a)$ – a set of player's actions, such as mouse click, double mouse click, and move (drag-and-drop). An object can be moved only from position to position.
- $R = (R_1, R_2, \ldots, R_u, ..., R_r)$ - a set of rules each under specific conditions oven F, for object type manipulation at specific position for given action; rules may differ from position to position and from action to action. For the drag-and-drop (move) action from position P_{source} to P_{dest} for object of type O_i (i=1..n), the rule could be Rule(Move, P_{source}, P_{dest}, O_i, T_t) = {if containsTyped(P_{source}, OT_k, T_t) >= 1 and containsAny(P_{dest}, T_t) < checkCapacity(P_{dest})} - e.g., for P_{dest} admissibility for allocation object of type OT_k over P_{dest} is checked, etc.), for the time step T_t. Note, that P_{source} and P_{dest} may be positions with indexes source and destination not defined as given integers but by arithmetic expressions, e.g. source mod 2 = 0 (for white positions at the chess board). There may be defined some restrictive rules forbidding moving an object from P_{source} to P_{dest}, providing certain conditions are met. to As well, for the action click $P_{source} \equiv P_{dest}$
- $E = (E_1, E_2, \ldots, E_f, ..., E_e)$ – a set of effects to be applied after each action over object of specific type residing given position: E_f = Effect(Action, P_{source}, P_{dest}, OT_t) may be a specific multimedia effect, possible increment of a game result, check of a position/object property, etc.
- 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 – game result as an arithmetic condition over F and attributes of P and O.

B. Sample Models of Quizzes as Board Games

Here, there will be presented several types of quizzes as board games and their models compliant to the model explained over. This model of quiz and puzzle games provides a powerful paradigm for facile and rapid construction of rich multimedia games of such types in a uniform way. The unification comes from the fact that educational games as quizzes, quests, puzzles and even instructional sets of e-learning activities can be represented as board mini-games. It may serve as a base for construction of graphical designer of games of such types with various levels of multimedia enrichment and complexity. The complexity of such a game will depend on the complexity of the rules and conditions and checking functions used for rule definition. Moreover, even more complex position board games as chess and backgammon can be defined by the model sketched over via function of neighborhood; however, they are not appropriate for educational purposes. For this reason, the examples of games designed by the authors and available in this section of the paper are in the area of quizzes, puzzles and quests.

Within a quiz, each mini-game will present a given quiz question. Unlike traditional quiz games, here the questions will appear in an appealing way, with advanced multimedia. Moreover, such board mini-games can represent not only quizzes but also specific logical problems to be solved. As well, they may present any set of teaching activities organizes in a list or set with random choice.

Next pictures show how some of the possible quiz questions are developed as board mini-game. Questions with one exact answer, e.g. "How many are the months in a year?" may be presented by a single position with title equal to the question and empty content editable through mouse click; after clicking over the position, an entry field will appear and the player will enter the answer. Next, as an effect from the entering the answer (as position content), the position content will be compared to the right answer. The final condition will check if the position content is not empty. Thus, there are only few of the game elements, as follows:

$PT = PT_1$

$P = P_1$, setType(P_1) = PT_1, setTitle(P_1, "How many are the months in a year?") = PT_1 PT

A = mouseClick, ENTERbutton}

$E = E_1$(openContent(P_1)), E_2(setContent(P_1, playerEntry))

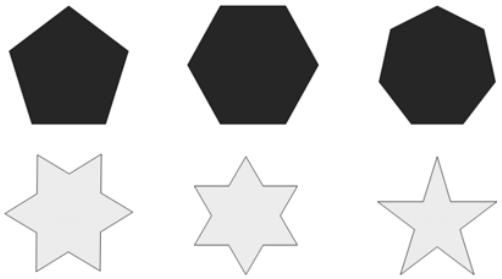
FC= {if checkContent(P_1) is not empty}

R = {R_1(mouseClick, P_1, E_1) – opens content entry field of P_1 for editing;

R_2(ENTERbutton, P_1, E_2) – closes content entry field and sets its value as content of P_1}

Res = {if checkContent(P_1)=("12" or "twelve") then result=true else result=false}

Questions 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 positional 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:



(a)

(b)

Fig. 1 Positional quiz board game with 1 to 1 mapping between position and objects at starting the game (a) and after game finish (b)

PT = Pentagon, Hexagon, Octagon, ServicePosition

P = P_1, P_2, P_3, P_4, P_5, P_6; setType(P_1, Pentagon), setType(P_2, Hexagon), setType(P_3, Octagon), set P_4, P_5, P_6 as ServicePosition's and invisible (used only for service purposes); all the capacities are equal to one.

OT = 5star, 6star, 8star

O = O_1, O_2, O_3, setType(O_1, 5star), setType(O_2, 6star), setType(O_3, 8star)

InitDisp – shown in table 2 below.

TABLE II
SAMPLE INITIAL DISPOSITION MATRIX

Object\Position	P_1	P_2	P_3	P_4	P_5	P_6
O_1						1
O_2				1		
O_3					1	

R = R_1(move, P_i, P_{ii}, O_j), for i=4..6, ii=1..3, j=1..3, apply E_1 and E_2

E = E_1=get(P_i, O_j, 1), E_2=put(P_{ii}, O_j, 1)

FC = {containsAny(P_4) = containsAny(P_5)= containsAny(P_6) = empty}

Res = {if (containsTyped(P_4, O_1)==1 and containsTyped(P_5, O_2)==1 and containsTyped(P_6, O_3)==1) then result=true else result=false} – fig. 1.b.

Classical image or word puzzles have easy construction process as board mini-games. The image building parts or the letters of a work puzzle are represented as objects, which should be moved o the right positions (wrong moves are possible). Logical quest may require action types different than move – such as simple and double clicks onto objects and object rotations, horizontal or vertical flips. Fig. 2 gives a view of a positional quest board game for matching symbols with minimum number of mouse clicks, for shortest time possible. The gamer has opened the first box containing a picture of a symbol (let say, a triangle), and now has to open a second such box (containing again a triangle) by a single click. While trying to match the other symbols, he or she has to remember where is located the second triangle. So, if the gamer will open it with a single click, both the positions with triangle object will stay opened with a melody played for the right match, otherwise they will both close with a disappointing noise and the number of wrong clicks will increase by one.

?	?	?	?
?	?	?	?
?	?	?	?

(a) Initial view at step 0

∩	■	?	Δ
?	Δ	?	?
?	?	?	∩

(b) Intermediate view at step k+1, (k+1) mod 2 = 1

∩	■	□	Δ
$	Δ	©	■
□	©	$	∩

(c) Final view of the game – all objects are visible

Fig. 2 Positional memory quest board game for matching symbols with minimum number of mouse clicks

Thus, the game helps mainly developing memory but if instead symbols there are placed for example musical instruments then it also makes gamers to remember form, name and timbre of most popular music instruments. The game model here is time-driven, as follows:

PT = '?' – there is only one position type with title='?'
P = P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}, P_{31}, P_{32}, P_{33}; setType(P_i, '?');
OT = ∩, ■, □, Δ, $, ∩, ©
O = O_1, O_2, O_3, O_4, O_5, O_6, O_7, O_8, O_9, O_{10}, O_{11}, O_{12}; setType(O_i, {∩, ■, □, Δ; $, Δ, ©, ■, □, ©, $, ∩} for i=1...12; setVisible(O_i, false) i=1...12
InitDisp – shown in fig.2.c
T = T_1, T_2, …, T_t, ..., T_{100} – the game may be played in maximum of 100 steps
A = mouseClick
E = E_1=Splash.wav, E_2=WellDone.wav, E_3=Wrong.wav
R = {R_1(mouseClick, P_i, O_j, T_k) = {delay(2sec); if (k mod 2)==0 then setVisible(O_j); E_1},
R_2(mouseClick, P_{ii}, O_{jj}, T_{k+1}) = {delay(2sec); if ((k+1)mod2)==1 then
if checkType(O_j)==checkType(O_{jj}) then result++; setVisible(O_{jj}); E_2
else setNotVisible(O_j); setVisible(O_{jj}); E_3
}
}
FC = {isVisible(O_j)==true for j=1…12}
R = value of result at step T_k

With similar design, there may be constructed board games with instructions of educational tasks such as finding the solution within a book, assembling a plant cell by its building blocks, searching and finding a treasure using a map and encrypted instructions, and many more.

C. Run Time Control of Quiz Board Game Execution

After proposing a new principal model for building educational quizzes, quests, puzzles as a sequence of board mini-games, it is time to consider the run time control of execution of such games. In general, it has to be applicable for a set of steps at moments T_1, T_2, …, T_t. The game engine should start from the initial disposition InitDisp and start listening for a player's action of one of the admissible actions types belonging to the set A. For each action the engine should apply one or all the possible rules if the needed pre-conditions are met. For each condition the result effect should happen, which may be a sequence of actions over attributes of positions and objects, getting/putting an object from/to certain position, increment of results, etc. Next, the finishing condition is checked – for example, the game may check if the END button (one of the positions) is pressed. When the game is over, the final result should be displayed.

In order to execute effectively each one of the steps in a time-driven game (the last example in previous section), the game engine should support a stack of the actions, applied rule(s) and effect for each step of the game. Thus, it would be possible to return to a step in the past, i.e. to make undo operations in a game until reaching certain step. In order to facilitate pause at a certain step and restart of the game from that step, the stack should include also all the current dispositions for all the steps passed. Without this information, when restarting from a certain time step, the game engine should start from the initial disposition and apply all the past rules until the last step, which may take some time and incurs a visible delay.

Finally, the game engine may select next question after the previous one is over. It is possible to use an adaptive strategy for selection of next question (problem) for the game, similar to the adaptive assessment. Like in [9], the adaptivity here is realized according student interests and knowledge level. For this purpose, questions have to be ranked at design time and, therefore, questions with greater ranks will give more points to the gamer than these of lower ranks. The game starts with very simple questions and, after correct answers, continues with questions of greater complexity.

IV. Adaptive E-learning with Boards Games

Main goal of adaptive e-learning systems is to improve e-learning processes by providing of different educational content which is the most appropriate for a particular learner or groups of learners. Some of these systems just personalized the training content according to user's preferences. Another of them beside personalization use more complicate criteria to adapt delivery of courseware such as learner knowledge, performance, goals and learning styles of students. The ADOPTA platform for building edutainment (education plus entertainment) services [16] is one such system. It delivers courseware in adaptable way depending learner style and knowledge result shown at assessment. For this purpose it uses adaptive navigation and adaptive content selection.

ADOPTA is a modular system (fig. 3), which includes an authoring tool for establishing the e-learning course content, instructor application for designing of adaptive courses and describing pedagogical approaches and software engine, which is responsible for adaptable content delivery to every individual learner. It makes use of learning styles to provide adaptive courseware delivery by means of adaptable navigation through the storyboard and adaptive content selection. Instructors can update on Internet navigation in narrative graphs and page content by a user-friendly Flash interface using drag-and-drop of learning objects and games from particular domain ontology. Content and link metadata are widely used in order to be used for controlling adaptive content selection and adaptive navigation.

ADOPTA is planned to make use of games of various types in order to select the proper game type for a given learner's character. Some of these types are planned to be the quizzes, puzzles and quests. As far as the model presented over and illustrated by several examples of games does support quizzes, puzzles and quests, authors plan to use it as a building paradigm for construction of a tool for designing such games. Moreover, the tool will allow construction of a more complex

problem solution game with involving within various teaching activities such as visiting some online course materials, instruction how to do some tasks, provisioning of auxiliary material, etc.

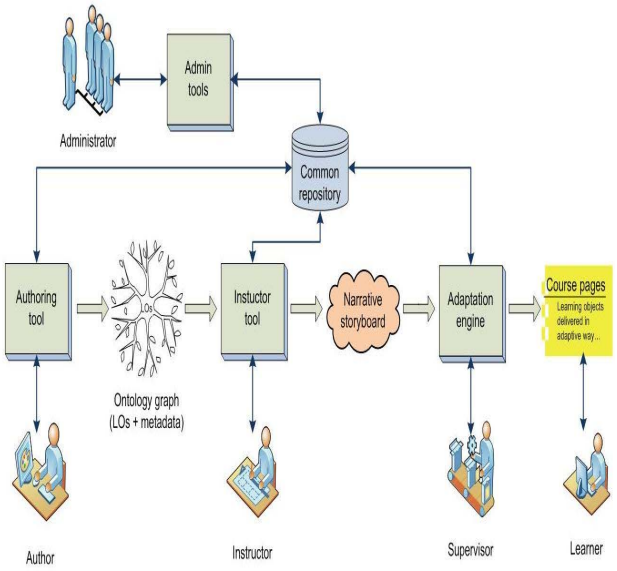


Fig. 3 General workflow of ADOPTA

In order to make courses more appealing and attractive to the students, quiz board games are going to be used intensively in ADOPTA. They will be designed by standalone game authoring tool using the model explained in section 3.1. Within the ADOPTA authoring tool, such games will be described as learning objects by a proper set of metadata showing the learning styles they are appropriate to be applied, the complexity level, game URL, and many other parameters. Next, instructors will allocate link to game URL at places they find to be appropriate and, finally, the adaptation engine will play the game. As far as each game presents a single quiz or a problem, the system should select automatically the next quiz question following the execution strategy mentioned in section 3.3.

V. Conclusions and Future Works

The paper has presented a new model of quiz questions and logical problems as multimedia board games. The model could be used for presentation of quizzes, mazes, quests, and other set of instructional activities suitable for e-learning. Following the model, various mini-games could be easily developed provided the following:

1. there is available a game framework for building such board mini-games, and,
2. there is constructed a game execution engine which runs the game model executing rules for each player's action under certain conditions.

The model offers a plenty of benefits which should be proven in real practice. First at all, it allows presentation of several important and popular types of educational games such as quizzes, puzzles, and quests. The model allows presentation of more complex logical problems which have to be solved by several actions delivered by the player according some rules and context conditions. Such resolution of logical problems is suitable for active online learning. Finally, a complex set of instructions and activities within given domain context can be represented as a board game using the same model. Examples for such scenarios are activities typical for visiting virtual museum, following a receipt for preparing a dish, locating and discovering an object using geographical map, and many more.

The future work will continue with designing a functional interface for game rule creation and, therefore, creation of an online tool for authoring and generation of Flex board games complaint with the proposed model. By means of ADOPTA instructor tool and using game metadata descriptions, such games will be incorporated within narrative storyboard and will be delivered to learners with appropriate profile (learning style, preferences, etc.).

Authors plan to make use of artificial intelligence agents - once as virtual opponent to the player within given game and, next, as virtual supporter of the gamer helping him/her in finding the right solution of the stated problem.

Acknowledgment

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

[1] Dempsey, J. et al. 1996, Instructional applications of computer games", *American Educational Research Association*, New York, 1996, pp. 8–12.

[2] Siang, A., Rao, R. 2003. Theories of learning: A computer game perspective. *Proc. of the IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE'03), pp.* 239-245.

[3] Prensky, M. 2006. *Don't bother me, mom, I'm learning!*, St. Paul, Minn., MN: Paragon House.

[4] Salen, K., & Zimmerman, E. 2003. *Rules of play: Game design fundamentals*. MIT Press, Cambridge, MA, USA, October 2003.

[5] Batson, L., S. Feinberg. 2006. Game Designs that Enhance Motivation and Learning for Teenagers, *Electronic Journal for the Integration of Technology in Education*, Vol. 5, pp. 34-43.

[6] Ferreira, A. et al. 2008. The common sense-based educational quiz game framework "What is it?", *ACM International Conference Proceeding Series*. Vol. 378, pp. 338-339.

[7] Bontchev, B., N. Gabarev, H. Pavlov. 2002. A Mobile Chess Game, *Proc. of 16th SAER Conference*, Varna, Sept. 2002, pp. 138-143.

[8] Feng, K. 2005. Joyce: A Multi-Player Game on One-on-one Digital Classroom Environment for Practicing Fractions, *Proc. of the Fifth IEEE Int. Conf. on Advanced Learning Technologies (ICALT'05),* pp. 543-544.

[9] Retalis, S. 2008. Creating Adaptive e-Learning Board Games for School Settings Using the ELG Environment, *J. of Universal Computer Science*, vol. 14, no. 17 (2008), pp. 2897-2908.

[10] Guetl, C. et al. 2005. Game-based E-Learning Applications of E-Tester. In P. Kommers & G. Richards (Eds.), Proc. of World Conf. on Educational Multimedia, Hypermedia and Telecommunications 2005, pp. 4912-4917.

[11] Dalziel, J. 2008. Using LAMS Version 2 for a game-based Learning Design, *Special issue on Comparing Educational Modelling Languages on the "Planet Game" Case Study, JIME*, November 2008.

[12] Hsiao, I-Han, S. Sosnovsky and P. Brusilovsky. 2009. Adaptive Navigation Support for Parameterized Questions in Object-Oriented Programming, *LNCS*, Vol. 5794/2009, ISBN 978-3-642-04635-3, pp. 88-98.

[13] Weck, O., Kim, I. Y., Hassan, R. 2005. Active Learning Games, *The 1st CDIO Annual Conference*, 06-09 June 2005, Kingston, Ontario. pp 1-15 (available at www.cdio.org).

[14] White, M. et al. 2004. ARCO - an architecture for digitization, management and presentation of virtual exhibitions, *Proc. of Computer Graphics Int. Conf.*, pp. 622-625.

[15] Bontchev, B., D. Vassileva. Modelling educational quizzes as board games, *Proc. of IADIS e-Society 2010 Conf*., ISBN: 978-972-8939-07-6, Porto, March, 2010, pp.20-26.

[16] Vassileva D., Bontchev B. "Adaptation engine construction based on formal rules", *Proc of CSEDU International Conference on Computer Supported Education*, 23 - 26, March, 2009, Lisbon, Portugal, INSTICC Press, Vol. I, ISBN: 978-989-8111-82-1, pp. 327 – 332.