Management of storyboard graphs for adaptive courseware construction

DESSISLAVA VASSILEVA, BOYAN BONTCHEV
Department of Software Engineering,
Sofia University, Faculty of Mathematics and Informatics,
5, J. Baurchier blv., Sofia 1164
BULGARIA
ddessy@fmi.uni-sofia.bg, bbontchev@fmi.uni-sofia.bg

Abstract: - Adaptive e-learning systems try to improve teaching processes on Internet by providing different educational content and activities for different learners. Modern tendencies of adaptation require usage of learning styles on student grouping. Main idea of adaptive e-learning applications is to use a different pedagogical approach for each of these groups. This paper presents a software module called *instructor tool* for creating training courses suitable for learners with different learning styles. This module is a part of a platform for building edutainment (education plus entertainment) services called ADOPTA (ADaptive technOlogy-enhanced Platform for eduTAinment). E-learning courses within the tool are presented by directed storyboard graphs. The course instructor can create and edit their graphs and pages by a user-friendly visual interface. Our article is aimed primarily at description of the graph management which is behind the visual presentation and specially at discovering working paths within these graphs.

Key-Words: - adaptive e-learning systems, adaptive hypermedia systems, instructor tool, storyboard graph, learning styles

1 Introduction

Adaptive Hypermedia Systems (AHS) are a part of adaptive software applications and in particular they are engaged with adaptation of hypermedia and hypertext. Most often these systems are web-based and can be used for the realization of applications in various areas such as adaptive e-learning, intelligent tutoring and adaptive games. This fact makes them very suitable for implementation of edutainment platforms such as ADOPTA (ADaptive technOlogy-enhanced Platform for eduTAinment) [1]. Their main goal in an e-learning context is to deliver hypertext and hypermedia content that is consistent with the profile of individual learner [2]. In order to achieve that aim, AHS use well known techniques, such as adaptive navigation, structural adaptation, adaptive presentation and historical adaptation [3]. Another trend in the adaptive system development is provisioning of tools for defining various pedagogical strategies for a course. Each one of them has to be appropriate for a particular learner in accordance with her/his learning style, knowledge, preferences and goals. Some of the adaptive e-learning systems that provide tools for instructional design are:

- AHA! ([4], [5]) - learning content is stored in pages, which are represented as XML files. They are composed of fragments that can be included conditionally and contain information about various concepts and their relationships. The presentation of the content page is determined at runtime and certain fragments are selected depending on the condition, which they are associated with. Concepts are defined only at the pages and there is no overall network reflecting relationships between them. This can lead to confusion and ambiguity among course authors.
- InterBook [6] - this application is one of the first AHS. It is not a complex and comprehensive system as far as provides only means for the creation and presentation of adaptive electronic textbooks. InterBook uses a simple network of concepts, modeling the relevant subject area. Each unit of an electronic book is associated with a set of concepts. The content of these units is static but relationships between them are created dynamically using knowledge gained from the network of concepts. Disadvantages of InterBook consist in the lack of support of advanced adaptive methods and insufficient set of interfaces.
- NetCoach [7] - knowledge of each training course is presented as a network of concepts.

Moreover, the system supports links between test questions and concepts. A set of test questions associated with a particular conception is used for assessment of learner knowledge about it. NetCoach does not support learning styles.

- ReCourse [8] – it is not an e-learning system but rather a tool for creating learning content in accordance with the IMS learning design standard. It is successor of the editor Reload and its main purpose is to facilitate the instructors of courses providing richer and user friendly interface.

In this paper, we present an instructor tool providing rich, comfortable and effective interface for creating courses including various pedagogical strategies. Unlike examined tools, this one supports learning styles of all kinds and is not bound to a specific style family. It is consistent with our principal conceptual model of adaptive AHS [9] as it is described in the next section. The instructor tool is a part of the ADOPTA platform for building edutainment (education plus entertainment) services [10]. ADOPTA is a modular system and includes: authoring tool for establishing the e-learning course content, instructor application (subject of this paper), and software engine, which is responsible for adaptable content delivery to every individual learner.

On one hand, providing user friendly interface is comfortable for instructors but on the other this hampers the implementation. Therefore, special attention is paid to the management of construction of a course as a directed graph of pages. The management algorithms can be used not only in the area of adaptive e-learning but also in other areas where graphs can be applied.

2 Conceptual model of AHS

The ADOPTA platform uses a new hierarchical organizational model which improves the AHAM reference model [11]. Table 1 represents the structure of its adaptability model of AHS together with explanation of the most important characteristics [9]. Main benefits of our conceptual model consist in assuring strong independence between learner profile, author content and pedagogical strategy [12], in support of different learning styles, content metadata, adaptive rule metadata and content packaging according the SCORM standard [13].

Our model has a hierarchical structure with two levels. At first level, the model is based on a precise separation between Learner, Domain and Adaptation sub-models, while at second level each of these sub-models is divided into three others sub-models. As shown in table 1, the Learner model describes profile of each learner such as her/his goals and preferences, knowledge and performance and learning styles. More specifically, the learning style sub-model defines for each learner her/his learning character including styles such as activist, theorist, reflector, pragmatist. This learning character can be polymorphic, which means that it is presented by order quadruple, since usually a particular learner is not fixed to a concrete style but rather to several ones at different level (fig. 1).

Learner Model - contains information for the learner profile. Depending on its meaning, it is stored in *Goals and Preferences*, *Learning Style* or *Knowledge and Performance* sub-models.	**Goals and Preferences**
	Learning Style
	Knowledge and Performance
Adaptation Model - includes description of each course storyboard graph (in *Narrative Storyboard* sub-model), metadata (such as link annotations, exam thresholds, etc.) of each narrative storyboard graph (in *Narrative Metadata* sub-model) and selection logic for passing over particular graph (in *Storyboard Rules* sub-model).	**Narrative Metadata**
	Narrative Storyboard
	Storyboard Rules
Domain Model - responsible for structuring of learning content. The content is granulized in LOs, which are connected each other within a relevant knowledge domain ontology. LOs and ontology are described by their metadata (in *Content Metadata* sub-model) respectively according IEEE LOM specification and Ontology Metadata Vocabulary (OMV) proposal	**Ontology graph**
	Learning objects
	Content Metadata
The **Adaptation Engine** communicates with each of the three sub-models at first level in order to generate and deliver to particular learner the most appropriate learning content for her/him	

Table 1: Tabular presentation of the structure of the conceptual model

The domain model contains structured learning content. It embraces three sub-models: learning objects (LOs) according to the SCORM standard, metadata about LOs and ontologies, and semantic ontologies organizing the content, i.e. the LOs. There are supported various types of LOs – not only narrative content but also additional types such as tasks, essays, assessment questions, games, etc. Each one of them could be associated with one or more narrative content LOs. The content LOs are developed by the author and next are placed by the course instructor on course pages.

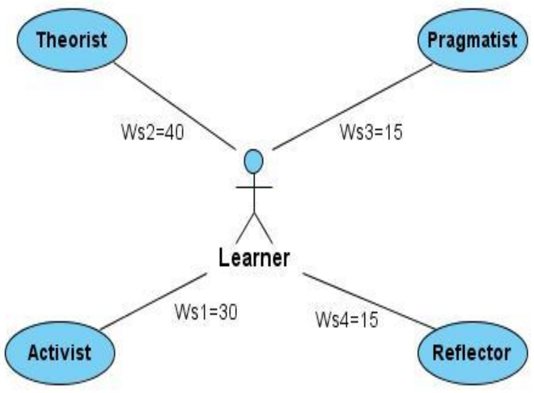

Fig.1: A sample learner character

The adaptation model (AM) takes key place in our model. It contains information about courses content, semantics of the pedagogical strategy employed by them and course organization. Courses are presented by so called narrative storyboard graphs.

Fig. 2 presents a sample for narrative storyboard course graph. Nodes of a storyboard course graph are either narrative pages (such as Page 1, Page 2) or control pages (CP) (such as ConrolPage 1 and ControlPage 2). Between any two CPs there are established so called work paths (WPs) consisting of content pages. Each one of these pages is composed of one or several LOs. For each of these LOs the instructor can assign a parameter that specifies conditions allowing LOs to be visible (for example, one such condition may be test results of a learner at a CP to be over a certain threshold). Information on these parameters' value is used by the adaptation engine in adaptive content delivery. Moreover, the instructor may define a weight of a WP for each learning style. Therefore, a particular WP may be appropriate for one or several learning styles. The adaptation engine determines which WP is most appropriate for a particular learner based on these weight and data from the Learner model. The control pages are used for assessment of current knowledge and performance for a learner, by automatic test generation. This test is composed of questions related to the LOs in the pages, which the learner has visited. The obtained assessment result is used for update of WP weights.

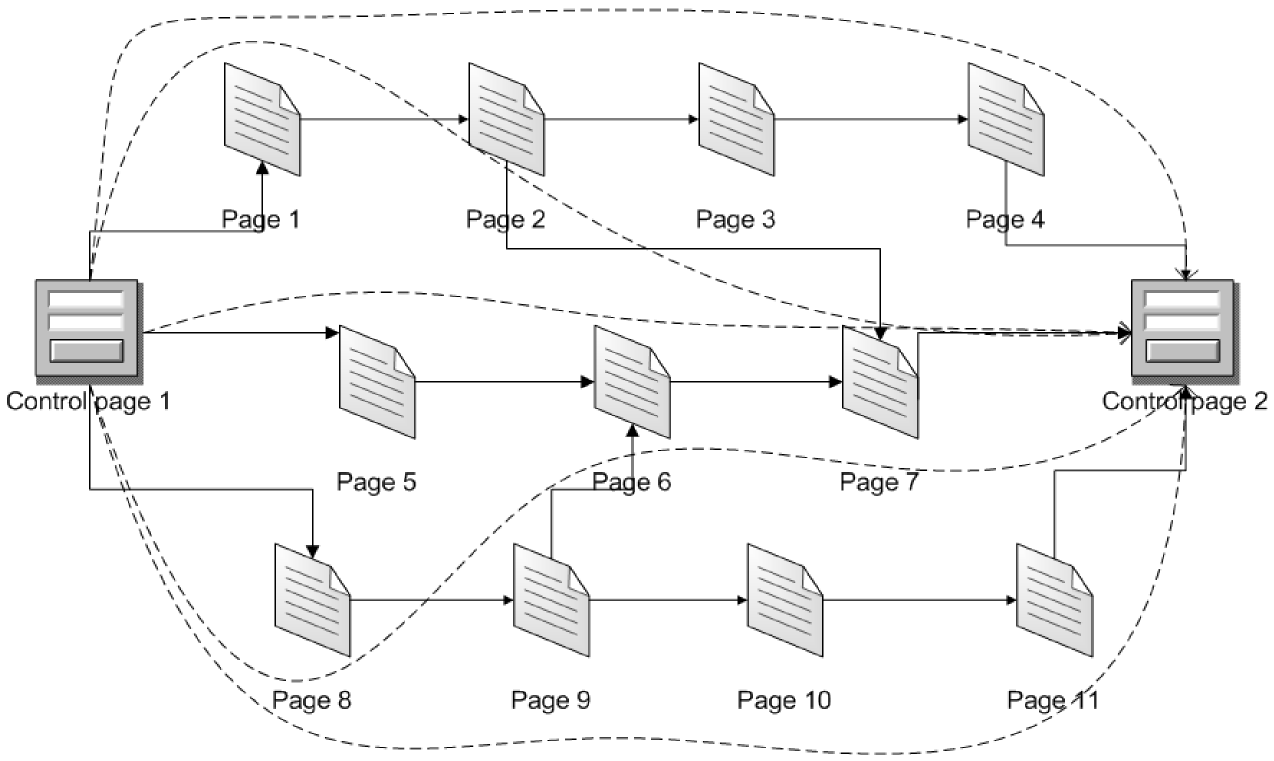

Fig.2: A sample narrative storyboard graph

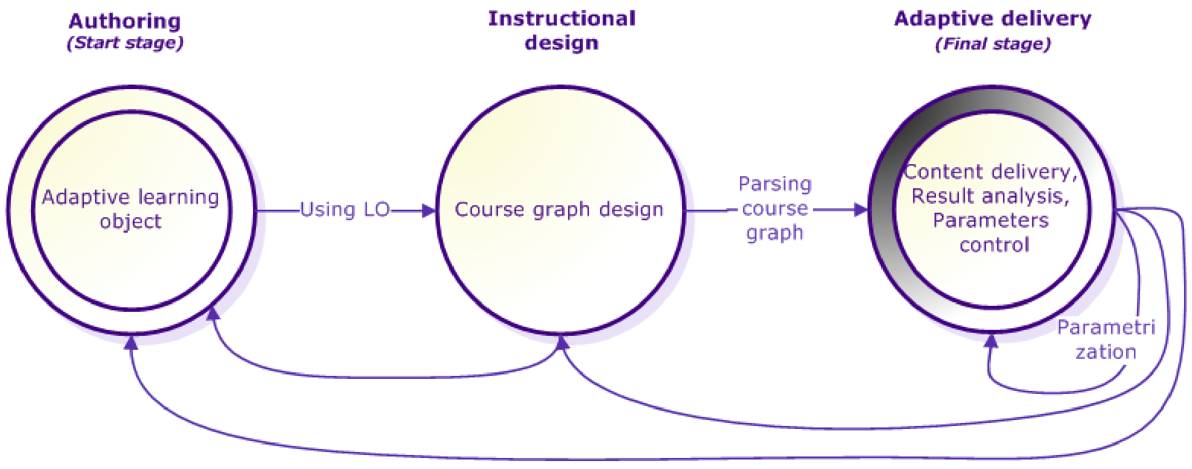

Fig. 3: The three stages in adaptive courseware design and delivery

3 Principal software platform architecture

3.1 General process workflow

Traditionally, there are three stages in design and delivery of adaptive courseware: authoring, instructional design and adaptive content delivery (fig. 3). The ADOPTA platform for adaptive e-learning includes an authoring tool, an instructor tool, an adaptive engine and a set of administration tools, all communicating through a common repository as shown in fig. 4. The content author is responsible for design of learning materials (objects) by organizing them within ontology with has-a and is-a relationships and, also, for metadata about LOs (by IEEE LOM) and about ontology itself (by OMV). The instructor uses the instructor tool to design a course as a narrative storyboard, by defining course pages and links between them. For a content page, he/she has to drag-and-drop at the page one or more learning objects from a proper ontology defined by an author. The supervisor is responsible for controlling the adaptation engine, e.g. for doing start and stop of adaptation behavior, tracking learner paths, etc. The administrator controls all the users using administrative tools.

Finally, the learner follows a course by receiving adaptive content and solving tests at control points. The learner is supposed to start at the first control point by filling an initial test about determining his/her learning style. Next, he/she follows the work path proposed by the adaptation engine but may opt to links to pages not belonging to the path and, thus, to divert to another work path. In such a case, they are always able to return back to the last visited page of the proposed path or, otherwise, to follow the new path until reaching a control point. There, the learner has to solve a test compiled by automatically selected questions about the LOs he/she has passed through.

3.2 Overall platform architecture

The ADOPTA software architecture is composed by three main layers – web clients, business layer and persistence layer, as shown in fig. 5. The persistence layer is presented by two sub-layers:

- Adopta Persistence Entities – ungrouped and common for all the platform applications
- Persistence Session Beans – grouped into specific and common, and used for read/store/edit of entities. Within this sub-layer, we have reused functionality for reading the same objects, while the business logic is specific for every module. Even in the case of login, UserEntity is always read but there are checked different roles.

The other layers are as follows:

- Business Java Session Beans – which are specific for each of the modules and contain its business logic
- Communication layer (Web services) – provide specific services for each of the modules.
- Web clients – represent web-based Flex service clients. Among its other benefits, this technology allows to generate easily web service client classes and method stubs. This is exactly the way the client consumes the published web services. The nature of the Flex applications to be run on the client side (browser/desktop) dispenses the application server with the load of rendering and manipulation the data.

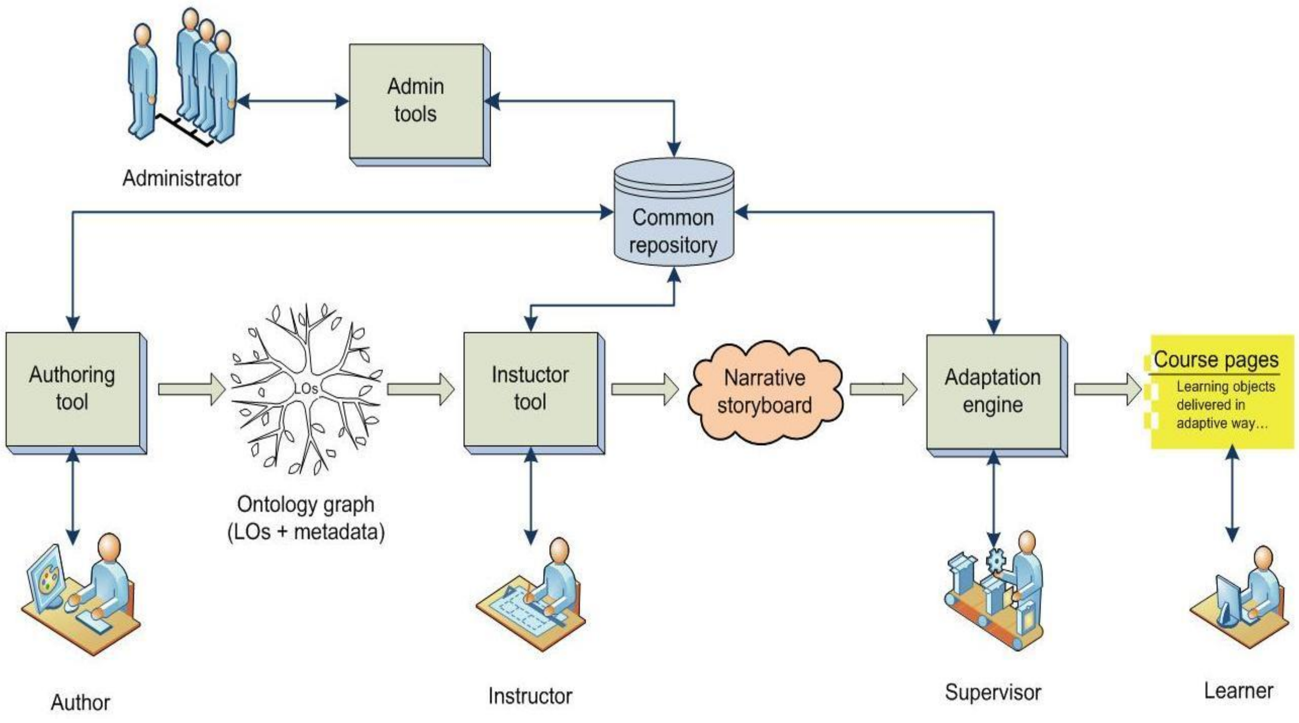

Fig. 4: View of the general workflow

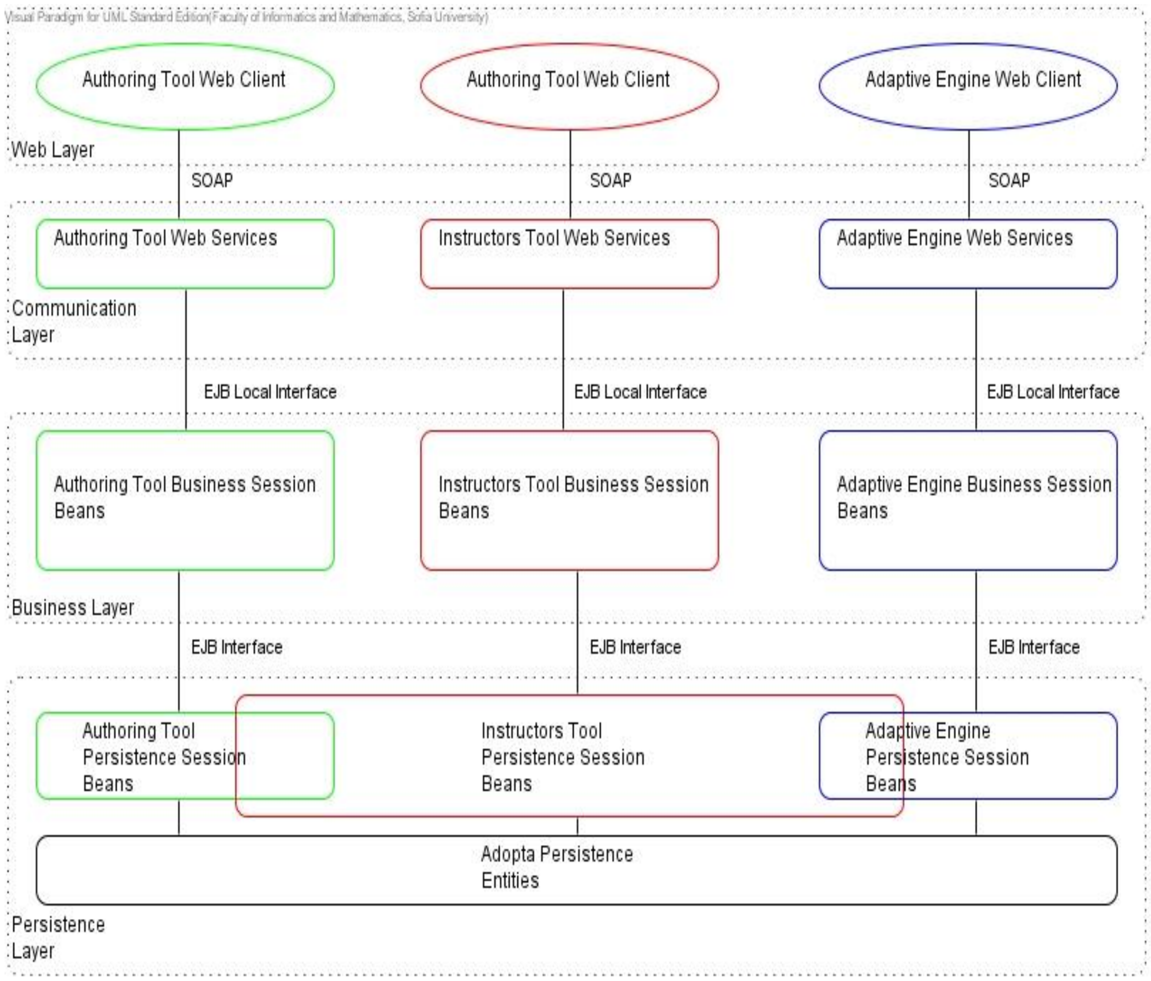
Visual Paradigm for UML Standard Edition(Faculty of Informatics and Mathematics, Sofia University)
Authoring Tool Web Client
Authoring Tool Web Client
Adaptive Engine Web Client
Web Layer
SOAP
SOAP
SOAP
Authoring Tool Web Services
Instructors Tool Web Services
Adaptive Engine Web Services
Communication Layer
EJB Local Interface
EJB Local Interface
EJB Local Interface
Authoring Tool Business Session Beans
Instructors Tool Business Session Beans
Adaptive Engine Business Session Beans
Business Layer
EJB Interface
EJB Interface
EJB Interface
Authoring Tool Persistence Session Beans
Instructors Tool Persistence Session Beans
Adaptive Engine Persistence Session Beans
Adopta Persistence Entities
Persistence Layer

Fig. 5: General platform architecture

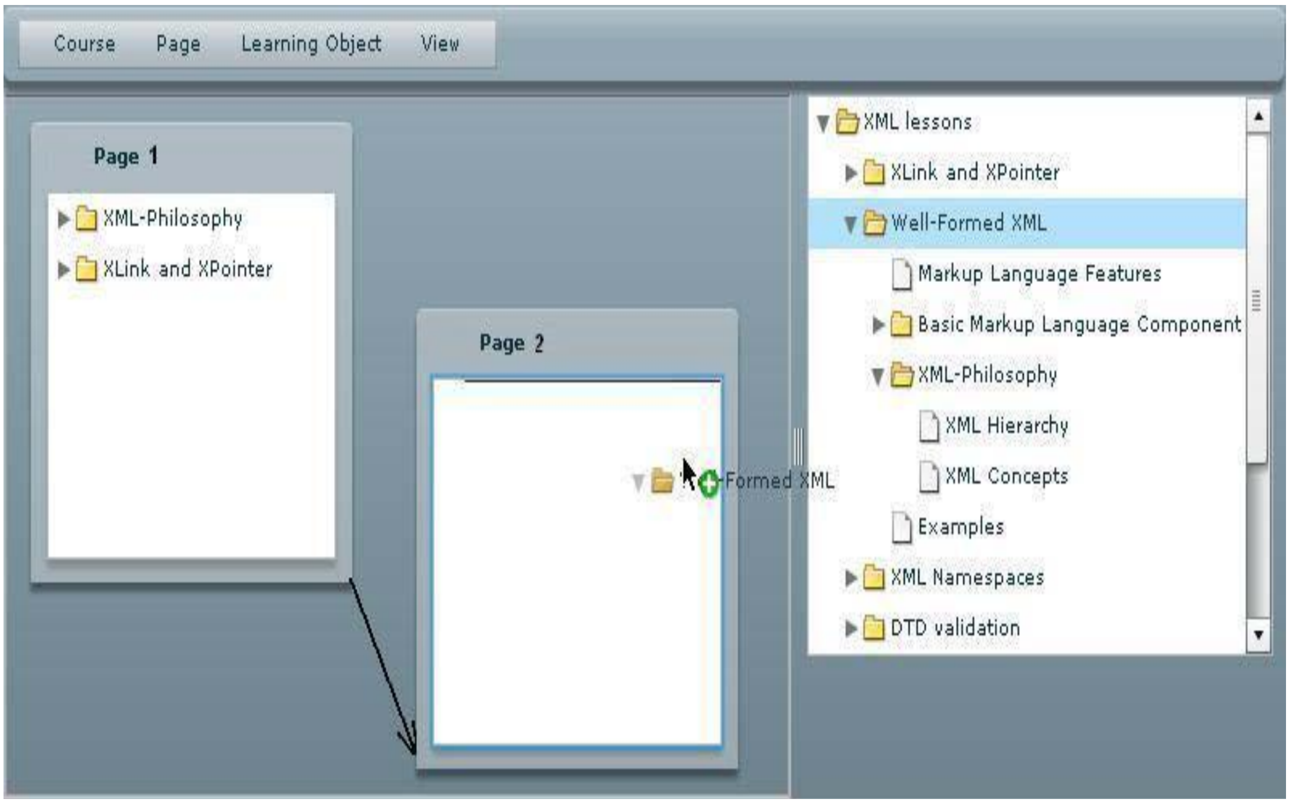

Fig. 6: View of the instructor application

4 Overview of the instructor tool

The instructor tool is a Flex application for creation via Internet of courses adaptable to different users with specific learning styles. The courses are composed in terms of interconnected pages represented as nodes of a narrative storyboard. The narrative storyboard graph is to be processed by the adaptation engine (AE) in order to choose the best working path for a particular user. Content pages can be easily modified by drag and drop of available learning objects. Fig. 6 shows instructors drag action from learning objects browser where they are organized in an ontology graph as defined by the author. In the course graph, there is one terminal vertex that represents the final control page, i.e. course exam. A course exam is generated automatically by choosing some of the questions related to the learning objects shown on pages of the work path leading to that CP (as far as questions are designed by the course author and linked to correspondent LO within the ontology tree). Thus, the instructor is not responsible for construction of assessment tests. To tune the course feedback, he/she can adjust CP thresholds values, i.e. level of assessment results for passed exam.

Instructors have also the responsibility to annotate page links and to set page weight parameters for each of the learning objects for given page. These page parameters are used for controlling the adaptive content selection and, therefore, are very important for tuning the system. The supervisor of AE may match parameters value to assessment result and, thus, he/she is able to control appearance of LOs for any particular learner. If the parameter of a LO within the page has high value and the learner has shown high performance at the last CP, this LO should be viewed to such a learner. Thus, when learner asks for the next page, adaptive engine may hide some objects that are not important for this user. Links annotation labels can be added also by instructor to influent user’s decision when a particular user is choosing among several links. If a learner abandons the work path determined by AE (by clicking on a link leading to another page outside of the path), the AE continues tracking pages the user has passed through giving the user ability to return back to the path by adding the link “Return back to the proposed path” to each of the pages.

The instructor uses a Web based client application developed in Adobe FLEX 3, as a rich internet application while the server-side of the application is developed in Java EE. Instructors may perform any action concerning creation and update of narrative storyboard including creating courses, creating pages, filling pages with learning objects, interconnecting pages, adjusting learning objects characteristics, setting link annotations, adjusting exam thresholds, and checking user feedback.

Thus, while editing narrative storyboard, the instructor has the responsibility to annotate page links and to set page weight parameters for each of the learning objects population the page. These page parameters are used for controlling the adaptive content selection.

5 Management of work paths

Within this section, we describe the management processes of creating and updating narrative graphs and work paths within these graphs. There are several actions concerning the discovery of working paths within the graph when changing the graph itself. Graph changes happen when creating and deleting a link between two graph nodes (content pages), and also when deleting a node.

When creating and editing narrative storyboard graphs, we cannot use well-known algorithms for discovering all paths between two nodes, because the instructor associates weight to each created path and supposes that this weight will be preserved at any change of this path. The path changes may occur when creating or deleting a link or a page (so called path change event) and then we have to preserve the weight for the updated path instead of asking the instructor again for the weight – this weight is going to be the same for all the path updates as far the instructor constructs the path according given strategy. This imposes the need of dynamic path discovery at any path change event.

5.1 Path discovery when creating links between pages

The work path discovery when creating links between pages (i.e., when the instructor adds a new arc from the current page to the next page) has several main cases:

- There are no outgoing arcs both from the current page and from the next page – if the current page does not participate in any WPs, then the tool creates a new WP containing current page and next page; else (e.g., in fig. 7 the current Page2 is included already in two working paths) we update paths containing the current page by appending next page to each of them.
- There are outgoing arcs from the current page but not from the next page (fig. 8) – in this case the tool checks which are the WPs including the current page (Page2 in fig. 8). Next, for each of these WPs it selects the list of pages from the start page of the WP to the current page and creates a new WP containing these pages and the next page if it does not already exist. For example, in fig. 8 for the WP=(Start page, Page1, Page2, Page2.1) we create a new WP=(Start page, Page1, Page2, Next page). However, for WP=(Start page, Page1, Page3, Page4, Page2, Page2.1) the tool does not create a new WP because such a WP already exists.
- There are outgoing arcs both from the current page and from the next page and no incoming arcs to the next page (fig. 9) – in this case the tool checks which are the WPs containing the current page (Page2 in fig. 9) and for each of them it selects the list of pages from the start page of the WP to the current page. Next, it adds these lists to the existing WPs starting from the next page and updates these WPs only if they do not already exist. E.g., in fig. 9 WP=(Start page, Page1, Page2, Page2.1) we select the list (Start page, Page1, Page2) and add it to the existing WP=(Next page, Page3, Page4) and WP=(Next page, Page6, Page7), i.e. we do update these two paths. However, when we treat WP=(Start page, Page1, Page2, Page2.2) we do no updates.

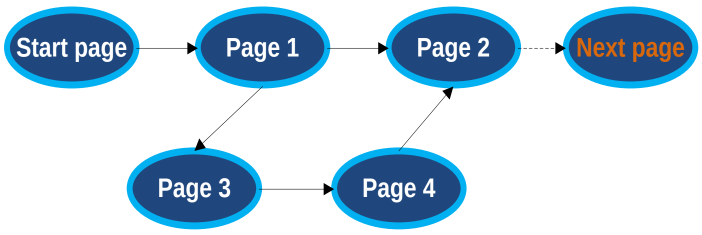

Fig. 7: No outgoing arcs both from the current page and from the next page

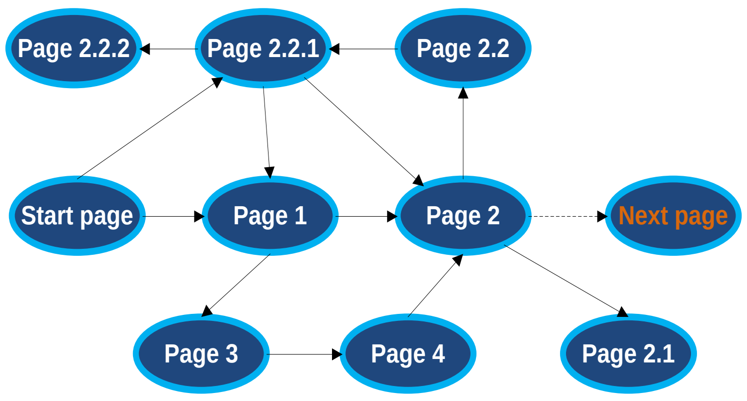

Fig. 8: Some outgoing arcs from the current page but not from the next page

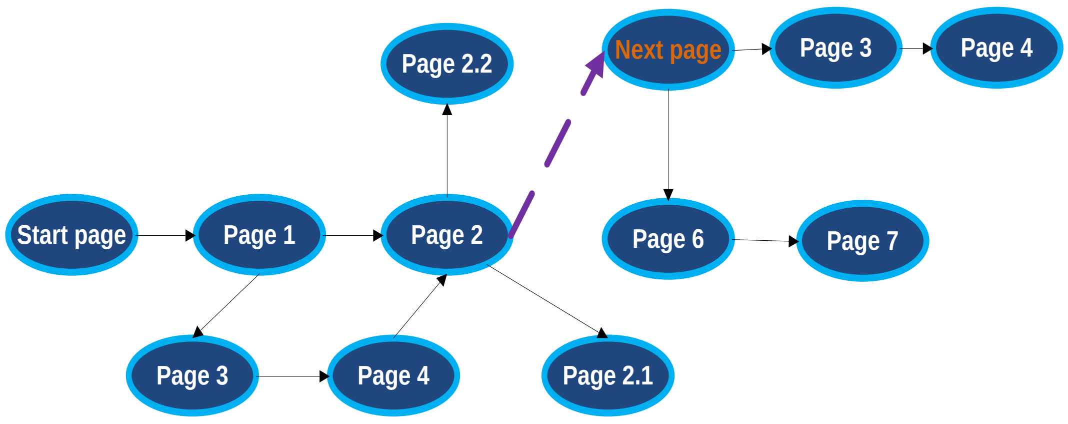

Fig. 9: Some outgoing arcs from both the current page and the next page and no incoming arcs to the next page

- There are outgoing arcs both from the current page and from the next page and some incoming arcs to the next page – here, the tool acts as in case (3) but does no updates of existing paths; instead of it there are created new WPs from the start page through both the current page and next page, until the pages finishing existing WPs where the next page participates.

5.2 Path discovery when deleting links between pages

During updating narrative storyboard graphs by means of the instructor tool, instructors normally delete content pages or links between content pages within the graph. So, when the instructor deletes an existing arc from the current page to the next page, the process discovering paths within the narrative storyboard graph has two main cases:

- There are no outgoing arcs from the next page – here, the tool selects all WPs containing current page (in the example given in fig. 10, this is Page2) and next page and deletes from them the next page. Moreover, if such a WP after arc deletion will contain only the current page, this WP will be deleted.
- There are some outgoing arcs from the next page (as shown in fig. 11) – in this case, the tool selects all WPs containing both current page (Page1 in fig. 11) and next page as neighbor pages. Next, it removes from these WPs all the pages after the next page and stores the resulted WPs if there are no repetitions. More, it creates new WPs starting from the next pages only if there are no incoming arcs to next page. For example, in fig. 11 the paths (Next page, Page5) and (Next page, Page6) will not be created. However, for the case in fig. 12 such paths will be created as far as there are no incoming arcs to the next page.

5.3 Path updates when deleting a page

When we delete a page, first the tool should do delete all the learning objects allocated on the page. Next, it deletes all incoming and outgoing arcs to and from this page according to the algorithm described in section 4.2. Thus, the case with path updates when deleting a page is a special case of discovering paths when deleting links between pages within the narrative storyboard graph.

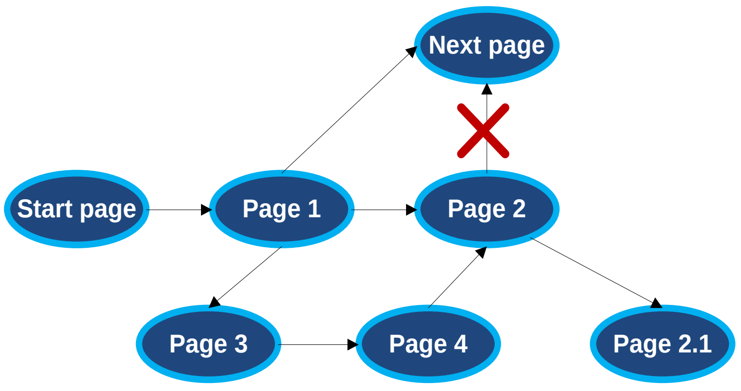

Fig. 10: No outgoing arcs from the next page

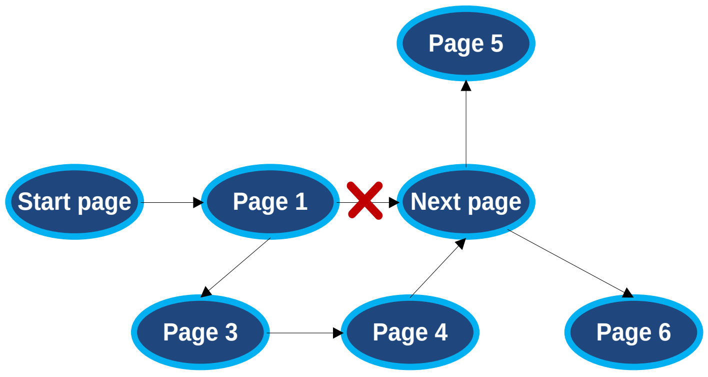

Fig. 11: Some outgoing arcs from the next page with incoming arcs

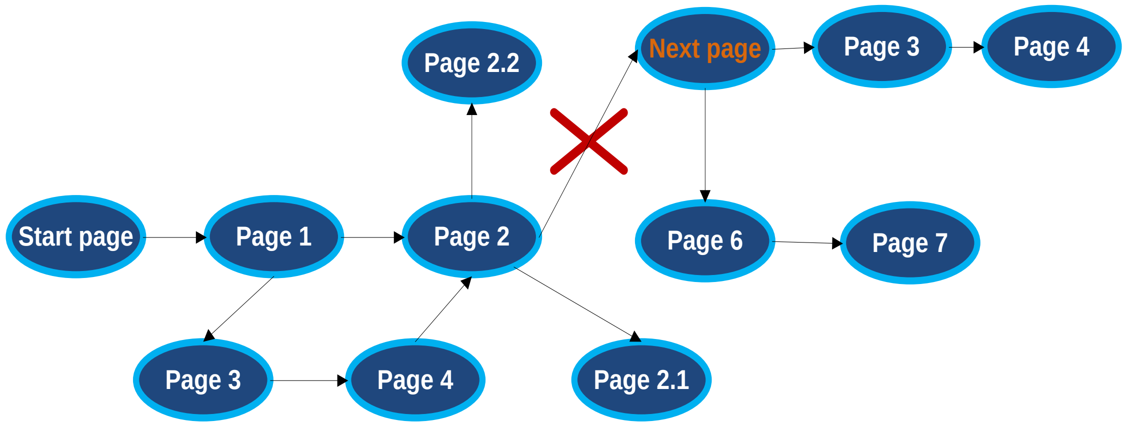

Fig. 12: Some outgoing arcs from the next page without incoming arcs

6 Conclusion

Adaptive hypermedia platforms have to be applied in order to improve traditional e-learning teaching process on Internet in a revolutionary new way. By using such systems, individual learners receive different educational content and activities according their particular learning characters. That is a serious reason to use learning styles for modeling the learner characters and, therefore, to apply different pedagogical approaches for each particular learner or a group of learners sharing the same learning character. ADOPTA (ADaptive technOlogy-enhanced Platform for eduTAinment) being under development at Sofia University [10, 12], Bulgaria, makes use of learning styles to provide adaptive courseware delivery by means of adaptable navigation through the storyboard and adaptive content selection. For reaching this goal, an instructor tool is constructed and integrated as a platform module in order to create and edit narrative storyboard graphs and content pages within these graphs. Thus, 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 from particular domain ontology. Content and link metadata are widely used in order to be utilized for controlling adaptive content selection and adaptive navigation, respectively [14].

A crucial issue of instructor tool design was the graph management module which acts dynamically behind the visual editor. It is used for discovering and updating of working paths within the graphs on-the-fly, at any event of graph modification. Dynamic path discovery and update is mandatory here as far as well-known algorithms for discovering all paths between two nodes cannot be used, because the instructor may need to associate weight to each newly created path at any time. Moreover, he/she supposes that this weight will be preserved next at any change of this path. Thus, for any path change event such as creating or deleting a link or a page we preserve the weight for the updated or/and newly created paths instead of asking the instructor again for the weight of these paths. In order to achieve this, the algorithms described over for dynamic path discovery have been applied and verified in practice.

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.

References:

[1] Bontchev B., Vassileva D. " Adaptive courseware design based on learner character", Proc. of *Int. Conf. on Interactive Computer Aided Learning (ICL2009)*, 23-25 Sept., 2009, Villach, Austria, pp.724-731.

[2] Dagger, D., Wade, V. & Conlan, O., Personalization for All: Making Adaptive Course Composition Easy, *Special issue of the Educational Technology and Society journal*, IEEE IFETS, 2005.

[3] De Bra, P., Brusilovsky, P., Conejo, R., Adaptive Hypermedia and Adaptive Web-Based Systems, *New York: Springer-Verlag*, 2002.

[4] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N., AHA! The Adaptive Hypermedia Architecture, *Publishing Proceedings of the ACM Hypertext Conference*, Nottingham, UK, August 2003, pp. 81-84.

[5] De Bra, P., Smits, D., Stash, N., Creating and Delivering Adaptive Courses with AHA!, *Proceedings of the first European Conference on Technology Enhanced Learning*, EC-TEL, Crete, October 1-4 2006, Springer LNCS 4227, pp. 21-33.

[6] Brusilovsky, P., Schwarz, E., Weber, G., A Tool for Developing Adaptive Electronic Textbooks on WWW, *Proceedings of WebNet'96 - World Conference of the Web Society*, October 16-19, 1996. San Francisco, CA, AACE, pp. 64-69.

[7] Weber, G., Hans-Christian, K., and Weibelzahl, S., Developing Adaptive Internet Based Courses with the Authoring System NetCoach, *In S. Reich, M. Tzagarakis and P. de Bra (Eds.), Book: Hypermedia: Openness, Structural Awareness, and Adaptivity, Lecture Notes in Computer Science LNAI 2266,* ISBN: 978-3-540-43293-7, pp. 226-238.

[8] Griffiths, D., Beauvoir, P., Liber, O., Barrett-Baxendale, M., From Reload to ReCourse: learning from IMS Learning Design implementations, *Distance Education*, Volume 30, Issue 2 August 2009, pp. 201-222.

[9] Vassileva D., Bontchev B., Self adaptive hypermedia navigation based on learner model characters, *Proc. of IADAT-e2006,* Barcelona, Spain, 2006, pp.46-52.

[10] Vassileva D., Bontchev B., Chavkova B., Mitev V. "Software Construction of an Authoring Tool for Adaptive E-learning Platform", Proc. of 4th Balkan Conference in Informatics (BCI'2009), 17-19 September, 2009, Thessaloniki, Greece, pp.187-192.

[11] De Bra P. at al., AHAM: A Dexter-based Reference Model for adaptive Hypermedia, *Proc. of the ACM Conference on Hypertext and Hypermedia,* 1999, pp. 147-156.

[12] Vassileva D., Bontchev B., Adaptation engine construction based on formal rules, *Proc. of CSEDU 2009*, ISBN 978-989-8111-82-1, Vol.1, Lisbon, Portugal, March 23-26, 2009, pp.327-332.

[13] Díaz, P., Sicilia, M.A. and Aedo, I., Evaluation of Hypermedia Educational Systems: Criteria and Imperfect Measures, *Proc. of the Int. Conf. on Computers in Education*, USA, 2002, pp. 621-626.

[14] Vassileva, D., B. Bontchev, S. Grigorov. Mastering Adaptive Hypermedia Courseware, *Acta Electrotechnica et Informatica*, Vol. 9, No. 1, 2009, pp. 57–62.