An Extendable Software Architecture for Personalized E-Learning systems

Omid Bushehrian
Department of Computer Engineering and IT,
Shiraz University of Technology
Shiraz, Iran

Robert Khaldar
Department of Electronically Educations,
Shiraz University
Shiraz, Iran

ABSTRACT
In this paper an extendable e-learning software architecture which supports personalized learning paths is presented. Any e-learning software which is designed based on this architecture can benefit from the loosely coupled interconnection among three main components of a personalized e-learning software: Workflow Management Engine (WFME), Recommender Component (RC) and Learner Interaction Component (LIC). By following this architecture the personalization method of the e-learning system which is implemented by RC component is very easy (open) to extend or change regarding the needs or strategies of a specific e-learning system. To achieve this, in this paper an FSP based method is proposed for designing a reusable WFME component. This component can be reused in any e-learning software which follows the proposed architecture.

General Terms
Software Engineering, Formal models, Extensible software design

Keywords
E-Learning systems, Workflow management system, Personalization, FSP language

1. INTRODUCTION
In recent years, e-learning has been used as a considerable web technology by educational institutes and learners are trained within this new educational style. Recently workflow technology is mostly employed as a design method for definition of learner's processes and learning paths of the courses within a learning environment precisely. Learners can follow different learning paths when workflow is used as a sequencing engine. Thus we can take advantage of effective procedural rules to define exact and flexible learning paths [2]. Usually, in a workflow based e-learning environment, the ones who take part are teachers and learners who interact in accordance with a predefined series of rules. In some e-learning systems which the learning procedure is not online, the workflow technology is very useful since it controls and guides the learners learning activities such as searching and studying a learning object, doing home-works and taking exams according to the learning paths and rules which previously have been defined by the course designer [3]. Once a workflow is defined, it can be executed and managed by deploying in workflow management system (WFMS) [2]. So the activities of learners are controlled by workflow engines in the learning processes [3].Furthermore, when composing an inter-disciplinary course from individual learning paths by a course designer, the priorities and quantities of some learning contents might be left undefined intentionally. Hence, the workflow management system, at the learning time, should also personalize the sequence and quantity of these learning contents for each learner based on his/her preferences. In order to achieve this goal, it is required that detail information about learners such as personal information, elective subjects, learner's characters, work sets and learning behavior are collected and analyzed. The learning behavior includes learner's progress, exercise, homework, test, question and so on.

Therefore it can be concluded that using the workflow technology in the context of e-learning systems entails at least these two challenges: (1) generating the workflows from the learning paths and learning precedence rules defined by the course designer and (2) designing the workflow management engine(WFME) with the extendable personalization support such that different personalization strategies and algorithms can be easily plugged into the WFME according to the needs and objectives of a specific learning environment.

In some previous works, formal languages have been chosen to specify the workflows. This approach not only provides the possibility of designing mechanical methods for translating learning paths and course precedence rules to the workflows but also enables the automatic verification of the resulting workflows. For example in [3] a formal definition method of learning paths using fuzzy Petri nets is presented. "Virtual Campus" [4], "Flex-eL" [1] and "ShareFast" [5] are other examples of using workflows in this context which also include personalized learning, i.e. each student can learn at his/her own preferred manner. However, none of these works have presented an automated method for producing formal workflows from teacher defined learning paths and rules as presented in our work.

In this paper a translation algorithm for translating the learning path ontologies [15] and the designer-defined precedence rules into workflows specified in FSP (Finite State Process) language [12], which is a formal language, is presented. The precedence rules are defined by the course designer once he intends to compose a new inter-disciplinary course from two or more independent courses (Learning paths). The main motivation behind choosing FSP language to specify the workflow has been the good potential of this formal language to specify and model different aspects of a comprehensive learning process including: precedence of learning contents, pre-defined precedence rules and constraints such as time limits on any learning content, defining course exams and their dominating rules, and most importantly, a good support of designing an extendable personalized learning software architecture.

Using the FSP as a formal modeling language to specify the workflows in our approach enabled us to propose extendable software architecture for personalized e-learning systems. In this

architecture, the Workflow Management Engine (WFME) is designed as a Labeled Transition System Analyzer (LTSA)[14] which interacts with a Recommender Component to support personalization. This engine itself can be easily plugged into any e-learning system that is based on this architecture as a backend, regardless of the frontend learning tools or facilities.

The remaining parts of this paper are organized as follows: The following section explains the components of our proposed e-learning architecture, in section 3 the main steps of creating an FSP-based workflow from learning path ontologies are described, the translation algorithm is presented in section 4, in section 5 the related works are presented and finally section 6 concludes the paper.

2. THE PROPOSED ARCHITECTURE

The proposed software architecture consists of three main components organized in layers: Learner Interaction (LI) Component, LTS Explorer (LTSE) Component and Recommender Component. The LI component has interactions with learners and "Learning Tools". "Learning Tools" supply functions and software packages for presenting learning contents, giving exercises, automatic question answering and online interactions. LI component repeatedly interacts with the LTSE component by calling the *NextAction* service to get the next action to follow according to the predefined learning paths (Stored in FSP-based workflow repository in Figure 1). The possible action types could be: starting a new learning content, giving an exam and continuing the previous action (for courses with time constraints). After LI component asked for the next action, it refers to the Learning Object Repository to retrieve the actual learning content and instructs the "Learning Tools" to load that learning content. As will be explained in the subsequent section, the workflow of each course is modeled as automata and stored in the "FSP-based Workflow Repository" in (Figure 1). At each state of the automata, a set of *Possible Actions* is detected by the LTSE component, once a call to the *NextAction* happens, and a "*Recommended Action*" is proposed by the Recommender component once the LTSE asks this component for one. This recommended action is selected from the possible actions considering the corresponding learner profile. The LTSE component requires some information about the learning outcome in order to prepare the list of Possible Actions at each step of the learning process. This information are passed, as parameters, to LTSE by the LI component once it is calling the NextAction(See Figure 1). The outcome of an exam (failed/pass) or the amount of time spent on a learning content, are examples of such information. In addition, the identifier of the learner for which the next action is requesting also should be passed to the LTSE component. The reason is that there might be several learners concurrently following their own learning paths and therefore the LTSE keeps track of their current learning states separately. Hence asking the next action from LTSE without telling him the requesting learner identifier would be meaningless.

In this architecture, the "Learner Profile Repository" is always kept up-to-date with the learner's new information. This information includes personal information, elective subjects, learning behavior, learner's progress and learning history which are collected by the LI component.

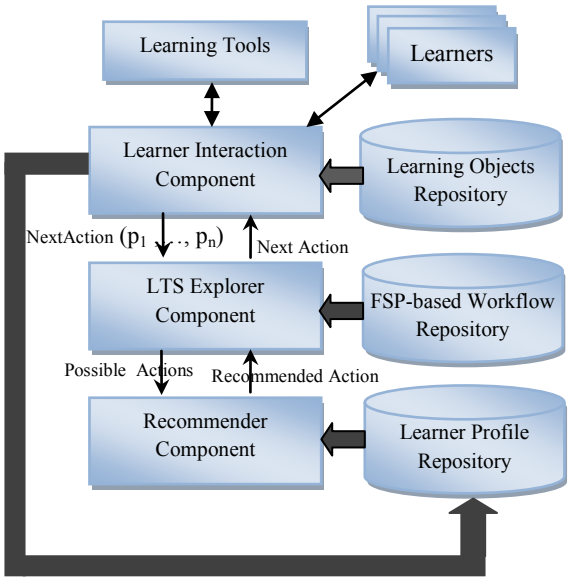
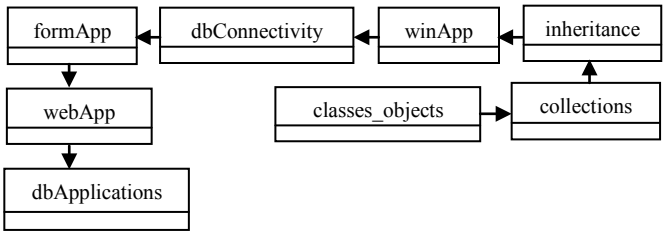


Figure 1.Architecture for WFMS

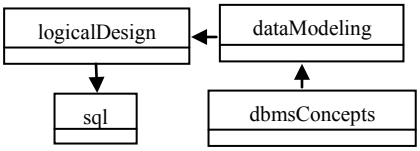
3. AN EXAMPLE

Learning path ontologies [15] are used to represent the learning objects and their prerequisite relationships in a course design with the goal of increasing the level of reusability of the design. In our approach the designer selects two or more courses and tries to design a new (inter-discipline) course by integrating their corresponding learning paths ontologies. To achieve this, the designer adds new rules to the existing prerequisite rules specified with the learning path ontologies of the selected courses. These rules indicate the prerequisite relationships among the learning objects of different courses which compose the newly defined course. For instance consider the following learning path ontologies in Figure 2. In this Figure part of the ontologies which represent the "is-prerequisite-for" relation among learning objects is shown for a composite course named "Practical .Net Database Programming". The workflow of this course is obtained by composing the learning paths of three courses: "OO programming in C#", "DB-Design" and "Programming Lab". The course designer also intends to add two tests: "OO test" and "DB design test" to the workflow.

Course1: OO programming in C#



Course2: Data-Base Design



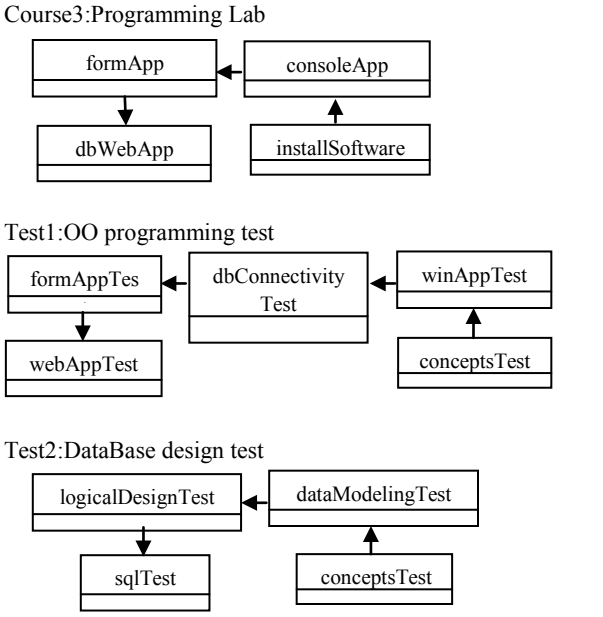


Figure 2. The learning path ontologies of the courses and tests

The designer also defines the following prerequisite rules among learning objects of the different courses and tests: "DBMS Concepts" from course2 must be taken before "DB Connectivity" from course1. "SQL" from course2 must be taken before "DB Applications" from course1. "Win App" from course1 must be taken before "Console App" from course3. "Form App" from course1 must be taken before "Form App" from course3. "DB Applications" from course1 must be taken before "DB Web App" from course3.

Also some orders and prerequisites are required for giving exams during the learning process for example within the first test the "classes_objects Test" should be given after finishing "classes_ objects" in the corresponding course and so on.

By adding the newly defined prerequisite relationships to the existing ones represented in the ontologies a Directed Acyclic Graph (DAG) corresponding to the new course is produced. In this graph the nodes represent the learning objects and the edges represent the prerequisite relationship. The resulting DAG then is converted to FSP (Finite State Process) by using a "Translation algorithm" and is stored in the "FSP-based Workflow Repository"(see Figure 3).

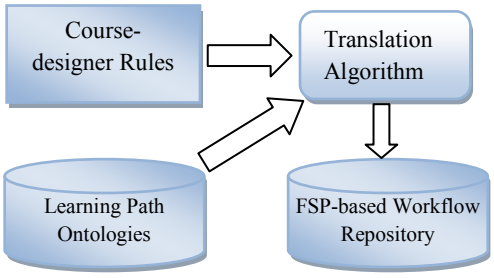


Figure 3.General stages

FSP's, are abstract machines used to model the behavior of concurrent systems [12]. A system modeled by FSP language is composed of a set of processes. Each process is an automaton which performs some actions repeatedly in known sequence. In this paper to model an inter-disciplinary composite course workflow, each individual Learning Path within the composite course is defined as a process, as shown in Figure 4.

The prerequisite rules among the Learning Objects (LO) across the different learning paths (defined by the course designer) are specified in the model by additional *synchronizer* processes called *Lock* processes. The translation algorithm is presented in the following sections. The resulting workflow is a *Labeled Transition System (LTS)* and the LTSE component is used to execute it. The LTSE component produces a list of possible actions at each step of the learning process and proposes one of them to the Learner Interaction component according to the learner's profile by collaborating with the Recommender component. For instance consider the composite course explained above. Considering all the prerequisite relationships in the workflow, the learner can start learning with any of the learning objects in the following set: {"classes_objects", "dbmsConcepts" ,"installSoftware"}. However the Recommender component may recommend the "classes_objects" to a learner with good knowledge of Data Base design course or interested mostly in object oriented design. To achieve better personalization, each learning object is associated with two properties: "learning duration" and "learning level". The Recommender component selects from the possible list of learning objects the one which is mostly compliant with the "learning strategy" defined for a specific learner. In the other words the recommendation is not only based on the previous learner knowledge or learning history in his profile, rather it is also based on the learning method or strategy which is associated with the learner profile. After selecting the learning object, the value of "learning duration" and "learning level" properties are set by the Recommender component according to the learning history of the learner and then it is passed to the Learner Interaction Component for presenting to the learner.

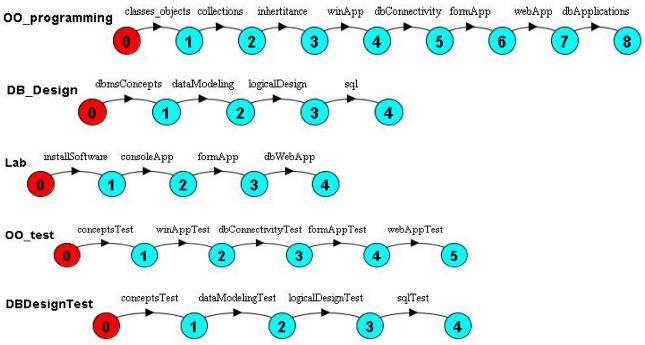


Figure 4. The learning paths corresponding to the three courses and two test represented as FSPs

Examples of different learning strategies are as follows:

- **No-Strategy:** There is no predefined learning strategy. The user selects his learning path himself.
- **Sequential:** The LO's of the same course are recommended consecutively to the end.

- **Shuffle**: The LO's from different courses are interleaved.
- **Quiz-Based Evaluation**: A test is selected as soon as it becomes available. There is no final exam.
- **Exam-based Evaluation**: The leaner is evaluated once at the end of a learning path.
- **Practical First:** The practical LO's are selected as soon as they are ready.
- **Theoretical First**: The theoretical LO's are selected as soon as they are ready.

The composition of the above strategies is also possible.

4. THE TRANSLATION ALGORITHM

As explained earlier, each learning path ontology corresponding to a course is represented as a DAG. The individual DAG's then are composed to form the composite DAG corresponding to the newly defined course. For example consider two DAG's of the courses C1 and C2 shown in Figure 5. The course designer adds a new prerequisite rule between learning object "d" from C1 and learning object "g" from C2 (shown by dashed line in Figure 6). The composite DAG is shown in Figure 6.

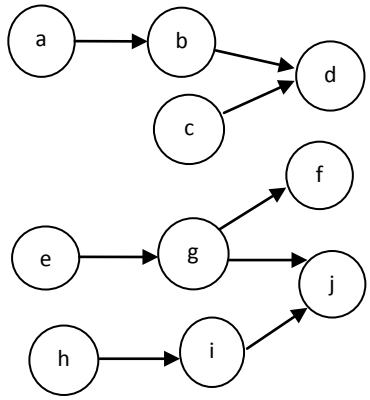


Figure 5. DAG's of courses "C1" and "C2"

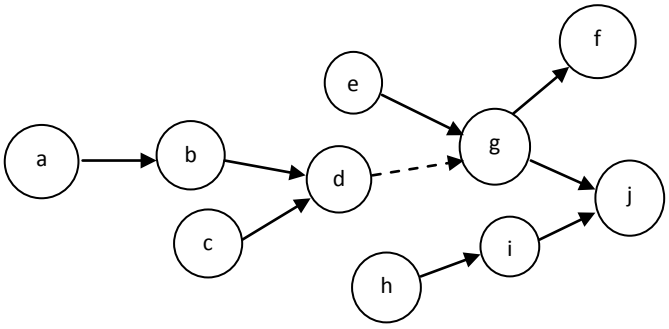


Figure 6. DAG of the composite course "C12"

At this stage the FSP's corresponding to the composite course should be generated. The main idea is to transform each learning path within the composite course in to a separate process. The precedence rules between LO's should also be considered. In the composite course C12, since "a" hasn't any prerequisite, a new process P1 is defined with the first action "a". Now we must decide for "b" which has only one prerequisite "a" which is written in existing process P1. So the "b" can be put after "a" in P1:

P1=($start_1$->a->b->STOP).

Since "c" has no prerequisite, like "a" it is considered as the starting action of a new learning path:

P2= ($start_2$ -> c -> STOP).

Likewise, for "h", "i" and e we have:

P3= ($start_3$ -> h -> i -> STOP).

P4= ($start_4$ -> e -> STOP).

"d" has two prerequisites: "b" and "c". In order to reach "d", it is needed to wait until "b" and "c" are visited. So a new process (P5) is required with action "d" however, the starting of P5 is synchronized with another "synchronizer" process named "Lock5". This process is used to control visiting "b" and "c" before "d":

P5= ($start_5$ -> d -> STOP).

Lock5= (l_{51} -> l_{52} -> u_5 -> STOP).

|| T5= (P1 || P2 || P5 || Lock5) /{ b/ l_{51} , c/ l_{52} , $start_5/u_5$ }.

Learning objects "g" and "j" are the same as "d" with two prerequisites. So two "Lock" processes are needed:

P6= ($start_6$ -> g -> STOP).

Lock6= (l_{61} -> l_{62} -> u_6 -> STOP).

|| T6= (P4 || P5 || P6 || Lock6) /{ d/ l_{61} , e/ l_{62} , start6/u_6 }.

P7= ($start_7$ -> j -> STOP).

Lock7= (l_{71} -> l_{72} -> u_7 -> STOP).

|| T7= (P3 || P6 || P7 || Lock7) /{ g/ l_{71} , i/ l_{72} , $start_7/u_7$}.

"f" has one prerequisite that is "g". A synchronization is needed to control that "f" is gotten after "g", Because "g" isn't last action in written processes and also course "j" can be gotten after "g".

P8= (start8 -> f -> STOP).

Lock8= (l8 -> u8 -> STOP).

|| T8= (P6 || P8 || Lock8) / {g/ l8 , start8/ u8}.

Finally the whole workflow is defined as follows:

||SYSTEM = (P1||P2||P3||P4||T5||T6||T7||T8).

The translation algorithm is presented here:

Algorithm DAG2FSP
INPUT DAG:D **OUTPUT** FSP: fsp
While (There is a not-visited node X_i in D) **do**

begin

1– IF X_i hasn't any prerequisite **THEN** create a new process like P_i :

P_i = ($start_i$ -> X_i ->

2– IF X_i has one prerequisite like Y_j **THEN** find process P_j that includes Y_j

2-1- IF Y_j is the last action of P_j **THEN** complete P_j :
$Pj = (\dots \rightarrow Yj \rightarrow Xi \rightarrow$
2-2- IF Yj is not the last action in Pj **THEN** create two new processes P_i and $Lock_i$:
$P_i = (start_i \rightarrow X_i \rightarrow \dots)$
$Lock_i = (L_i \rightarrow U_i \rightarrow stop)$
[L_i is synchronized with Yj and U_i is synchronized with $start_i$]

3 – IF X_i has two or more prerequisites ($Y_1 \dots Y_k$) which all of them must be visited before X_i ("AND" constraint) **THEN** create two new processes: P_i and $Lock_i$:

$P_i = (start_i \rightarrow X_i \rightarrow \dots)$

$Lock_i = (L_{i1} \rightarrow L_{i2} \rightarrow \dots \rightarrow L_{ik} \rightarrow U_i \rightarrow stop)$

[L_{i1} is synchronized with Y_1, L_{i2} is synchronized with Y_2, … – L_{ik} is synchronized with Y_k , U_i is synchronized with $start_i$]

4 – IF X_i has two or more prerequisites ($Y_1 \dots Y_k$) which visiting one of them is adequate to start X_i ("OR" constraint) **THEN** create two new processes: P_i and $Lock_i$:

$P_i = (start_i \rightarrow X_i \rightarrow \dots)$

$Lock_i = (L_i \rightarrow U_i \rightarrow stop)$

[L_i is synchronized with $Y_1 \dots Y_k$, U_i is synchronized with $start_i$]

5. RELATED WORKS

Some popular products, such as WebCT Vista [6], WBT TopClass [7], Lotus Learning Management System [8], Oracle iLearning [9], BlackBoard Learning System [10] allow assembling several different courses to build a new one but still don't fully consider giving teachers opportunities to define a structural course for guiding students to navigate learning resources.

The IBM Lotus LMS[8] which is the most interesting products from the above list supports both the course designing by providing the authoring tool and personalized learning. The authoring tool provided with the LMS is a course planning, development, and packaging application. The sole purpose of the Authoring Tool is to create Web-based content for the LMS. In our proposed architecture the course designing is supported by a mechanical translation algorithm which translates learning paths ontologies to FSP. This is very useful when the designer intends to design an inter-disciplinary course.

IBM Lotus LMS, uses the user profiles to represent skill sets and areas of interest, and to follow a personalized learning paths. The LMS can track a student's personalized learning path and can use the profile information to sequence learning objects within the course. Sequencing is based on learning objects prerequisites, objectives, or evaluation scores. IBM Lotus LMS doesn't support Workflow theoretical foundations, while our proposed architecture uses the FSP as a formal modeling language to specify the workflows so it enables us to propose extendable software architecture for personalized e-learning systems.

Workflow technology is recently used to design and develop process-centric courses and monitor student process like Virtual Campus [2], Flex-eL [1]. In Virtual Campus which is a research project sponsored by Microsoft Research, courses are defined as workflows. By doing this, we can exploit powerful procedural rules in order to define precise while flexible learning paths. The obtained e-learning system is a combination of process definitions (schedules in the BizTalk [4] jargon), Web Services, BizTalk code plug-ins, static web pages, frames and dynamic web pages. The internal code plug-ins have been written using the C# language under the Microsoft .NET framework, using the provided BizTalk API. The internal code plug-ins have been mainly used to allow the interaction of the students with the WFMS. Flex-eL[1] utilized workflow technology to facilitate personalized activities sequencing by using the structural aspect of workflow specification to model online courses. A process-modeling tool called *FlowMake* is used to capture the study process. The course activities and associated roles are identified and modeled using the tool. Exporting the process model in to a workflow repository from *FlowMake* is also possible. The workflow model is then deployed in the workflow server which has been built upon Microsoft SQL server 2000.

In [3] a formal definition method of workflow based courses using fuzzy Petri nets and a workflow based e-learning architecture, called WELA, is proposed. However in this paper the course designer is not supported by the automated tools for designing a course from high-level learning path ontologies. Besides, the proposed architecture in this paper is not designed extendable.

Paper [5] is proposed a new educational framework using an e-learning system called *ShareFast* which is a Semantic Web-based software for document management system with workflow. It can also record learner's input and output history data for the instructor to conduct performance analysis activities. ShareFast, has been developed by the members of Design Engineering Laboratory, the University of Tokyo. It is an open source, client/server application for document management based on workflow using RDF metadata on Jena framework[16]. In [13] an architecture for personalized e-learning system is presented based on an intelligent agent. In order to to form a personalized interaction to learner, learning tools, managing tools, assistant teaching tools and intelligent agent assistant are collaborating within an integrated e-learning environment. The fuzzy matching and decision tree techniques are used to design the personalization feature. However this system is not based on a formal workflow system as presented in our work.

6. DISCUSSION

Table 1 summarizes a comparison between our proposed e-learning architecture with previous ones based on seven criteria. The *Architecture extendibility* refers to the property of easy replacing an implementation of a component with a new one. The *Separation of concerns* refers to the principle of designing the e-learning system concepts and components as independent as possible. It increases the reusability and ease of change of the designed components. For example in our architecture, the learning paths, the course evaluation strategies and the personalization methods are defined as separate concepts as follows. Learning paths and evaluation learning objects are defined as separate processes in FSP language. The actual evaluation is performed by Learning Tools and only the result is passed to the WFMS. The personalization procedures are also implemented by the Recommender Component. The *Ease of implementation* refers to the availability of platforms or engines for executing the course workflows. Many previous works use the third-party commercial work flow management systems. The work flow management system of the proposed architecture is based on the well-known Labeled Transition Systems (LTS) and

hence the implementation is very straightforward. The *Personalization support* indicates to what extent different aspects of the personalization is supported within the e-learning system. Our proposed architecture only supports the learning object election and learning duration regarding the learner previous learning history.

Table 1.Comparison between our proposed e-learning architecture with previous ones

	WELA[3]	IBM lotus LMS[8]	Agent-based personalization Architecture[13]	The proposed architecture
Architecture extendibility	NO	NO	Yes/Due to using separate modules for introducing strategies	Yes/A personalization strategy could be replaced with a new one.
Separation of concerns	No/ The definition of Learning paths , evaluation strategies and personalization methods are not separated	Yes/By using SCORM	Yes/Because of using different modules for each part	Yes/Learning paths, evaluation strategies and personalization are defined separately.
Ease of implementation Of the selected Formal model	Yes/ Automatic translation to BPEL4ws	N/A	N/A	Yes/ WFMS is based on well-known Labeled Transition Systems(LTS)
Personalization support	Partially /based on past learner experiences and knowledge	Partially/based on skill sets and areas of interest	Yes/ personalization is based on intelligent agent and human roles influence results.	Partially /based on past learner experiences and knowledge
Workflow theoretical foundation	Yes/Petri Nets	NO	NO	Yes/ FSP
Course designer support	No	Yes/Course designers and developers can use the Authoring Tool	Yes/ It is done by assistant teaching tools	Yes/Automatic translation of Learning Path ontologies to FSP

Other aspects of the personalization which are related to the methods of presenting a learning object to the learner (suitable to his learning behavior), is delegated to the Learner Interaction component in our architecture. The *Theoretical foundation* factor means that whether the workflow is specified in a formal language or not. In this work the FSP language is applied to specify learning paths, evaluation learning objects and precedence rules formally. Finally, the *Course-designer support* refers to the automated tools that help the course designer to design the courses and learning paths. For example, by applying the translation algorithm presented in this paper, the course designer is needless of dealing with the FSP language syntax and semantics.

7. CONCLUSIONS

In this paper a new e-learning architecture based on the FSP language was presented. The workflow management system of this architecture was designed as a Labeled Transition System analyzer. Designing an e-learning software in terms of components and organizing them in layers with standard interfaces brought the reusability and extendibility properties to the architecture. The extendibility feature particularly is very important for personalization purpose because different e-learning systems may prefer different personalization strategies.

8. REFERENCES

[1]. Sadiq, Sh., Sadiq, W., & Orlowska, M.(2002). Workflow Driven e-Learning: Beyond Collaborative Environments, School of Computer Science and Electrical Engineering, Distributed Systems Technology Center ,The University of Queensland, QLD 4072 Australia

[2]. Cesarini, M., Monga, M., & Tedesco, R.(2004). Carrying on the E-learning Process with a Workflow Management Engine, In the Proceedings of the 2004 ACM symposium on Applied computing (SAC '04), pp. 940–945, New York, NY, USA, :ACM Press

[3]. Kong, W., Luo, J., & Zhang, T.(2005). A Workflow based E-learning Architecture in Service Environment, In the fifth International Conference on Computer and Information Technology(CIT05):IEEE

[4]. Microsoft biztalk, http://www.biztalk.org/

[5]. Hiekata, K., Yamato, H., Rojanakamolsan, P., & Oishi, W.(2007). A Framework for Design Engineering Education with Workflow-based e-Learning System, JOURNAL OF SOFTWARE, VOL. 2, NO. 4: ACADEMY Publisher

[6]. WebCT. Vista, http://www.webct.com

[7]. WBT. Topclass e-learning suite, http://www.wbtsystems.com

[8]. IBM Lotus learning management system, http://www-01.ibm.com/software/lotus/products/learning-management-system

[9]. Oracle ilearning, http://iLearning.oracle.com

[10]. Blackboard, http://www.blackboard.com

[11]. Ghaleb, F., Daoud, S., Hasna, A., ALJa'am, J.M., El-Seoud, S.A., & El-Sofany, H.(2006). E-Learning Model Based On Semantic Web Technology, International Journal of Computing & Information Sciences, Vol. 4, No. 2

[12]. Magee, J., & Kramer, J.(1999). Concurrency:State Models and Java Programs , Chichester, England: John Wiley and Sons

[13]. Li, W., & Li, X.(2009). Design of a Personalized Learning System Based on Intelligent Agent for E-learning, Ninth International Conference on Hybrid Intelligent Systems, pp. 187-190, Shenyang: IEEE

[14]. Ayles, T., Field, A.J., & Magee, J.N.(2003). Adding performance evaluation to the LTSA tool, 13th Int. Conference on Computer Performance Evaluation: Modeling, Techniques and Tools, Lecture Notes in Computer Science, LNCS 2794: Springer

[15]. Pirrone, R., Pilato, G., Rizzo, R., & Russo, G.(2005). Learning Path Generation by Domain Ontology Transformation, Advances in Artificial Intelligence, Vol. 3673, pp. 359-369

[16]. Jena: A Semantic Web Framework for Java, http://jena.sourceforge.net/