A Personalized e-Learning System Based on EGL

Jie Gao and Bo Song

***Abstract*—With the development of Internet technology, information resource is being spread widely, thus the issue of information overload formed. It becomes more difficult for users to retrieve their needed information from so enormous information space. To solve the issue of information overload, recommender system emerges as the times require. This paper presents the personalized e-learning system based on Collaborative Filtering Recommender Algorithm. And the system will be implemented based on EGL with the advantages of developers paying attention to the business issues without caring for software technical details. By the solution of EGL, the function of personalized information recommendation of e-learning system will be achieved and personalized information will be recommended to users to help them improve learning efficiency. The results show that the solution of EGL is more flexible and simple, reduce the development cycle and cost, improved the real-time requirement of e-learning system.**

***Index Terms*—EGL, recommender system, personalized, e-learning, architecture.**

I. Introduction

With the development of Internet technology, huge information resource is presented to us constantly and is being spread widely. The scale of information resource in the internet is also growing, thus the issue of information overload formed. It becomes more difficult for users to browse all of the information resource and find out what they need or they are interested in. Facing information overload issue, academic cycles have carried out a number of researches and proposed many solutions for personalized information. Recommender system is one of the solutions. It is an intelligent and personalized information service system and describes the user's lone-term information need by user modeling, based on which it can customize the personalized information with the specific recommendation strategy [1]. Collaborative Filtering is one of the most popular approaches for determining recommendations [2]. It can be divided into two parts. One is based on users' preceding ratings by calculating the similarities among users. The other one implements recommendation by calculating the similarities among items. In this paper, we will construct a personalized e-learning system based on Collaborative Filtering Recommendation Algorithm and will implement the system based on EGL (Enterprise Generation Language).

Because the e-learning system can provide richer end-user experience, RIA architecture has been adopted by more and more developers. For e-learning system, the main benefit of Ajax is a greatly improved user experience. Although JavaScript and DHTML – the technical foundations of Ajax – have been available for years, most programmers ignored them because they were difficult to master. Although most of the Ajax frameworks available today simplify development work, developers still need a good grasp of the technology stack. So, if you're planning to use Ajax to improve only your application's user experience – if you're not also using it as a strategic advantage for your business – it may be unwise to spend a lot of money and time on the technology [3]. ORM (Object-Relational Mapping) is a kind of technology which can solve the problem of impedance mismatch between Object-Oriented Programming and RDB (Relational Database). EJB 3, Hibernate and Oracle TopLink are the effective solutions to implement ORM, but the implementation of ORM is time-consuming compared with JDBC [4]. Although the foregoing ORM tools provide convenience for operating RDB as object, the cost and complexity of e-learning system are increased, and the real-time requirement of e-learning system is reduced [5].

Aiming at the above-mentioned problems, so the application architecture of e-learning system in this paper is implemented based on EGL, which is a high-level language and lets developers create business software without requiring that they have a detailed knowledge of runtime technologies or object-oriented programming. So the key feature of the architecture based on EGL – developers can focus on the business issues what code handle without caring for software technical details – is implemented by using existing platform and technology instead of replacing them and reduce the cost and development cycle and improve the real-time requirement of e-learning system.

II. EGL and ExternalType

EGL language is architected to reflect patterns that are common to different kinds of business software and hides many details that are platform specific. EGL also helps a company retain developers who are knowledgeable in business processes, even if those developers lack the time needed to stay current with technical change. And the relative simplicity of the language helps traditional developers become accustomed to the latest technologies.

The rational products that support EGL are based on Eclipse, which is called EDT (EGL Development Tools). EDT includes the core language packages – EGL SDK and corresponding IDE(integrated development environment) [6]. In the EDT, EGL is used in a development process that has defined three steps. The first stage is to compile the EGL code. EGL compiler will scan and analyze the syntax and semantic

Manuscript received April 1, 2014; revised May 16, 2014. This work was supported by the Science and Technology Project of Education Department of Liaoning Province, China (Research of e-Learning System of small and medium-size Enterprises Based on SaaS, No. L2013417).

The authors are with Software College, Shenyang Normal University, Shenyang, Liaoning, China (corresponding author: Song Bo; e-mail: songbo63@aliyun.com, gaojiexy@126.com).

of EGL source file, then generate the IR (intermediate representation). The second stage is code generation. The code generator read and parses the IR (eglxml file) and then generates corresponding target language (Java or JavaScript). The third stage is to use target language compiler to compile into executable code. The Java compiler will compile the EGL-generated Java code into class files to run in different platforms and the EGL-generated JavaScript code will be interpreted in the browser environment.

As is shown in the compilation process of EGL, EGL itself does not run directly and it must be compiled and executed by the compiler of target language to build and generate target language on corresponding platform. And in the process of language generation, different platforms and existing technology can be integrated and made full use of. EGL is not to replace existing technology and not to unify to develop new language and it is to maximize the use of mature technology as well as the supplement and extension of existing technology.

In an EGL application, serial user-defined logic parts are provided, such as Handler, Program, Library, Service, ExternalType and so on. These logic parts constitute an EGL application jointly and form the workflow of application. Among them, Handler, Program, Service and Library belong to generatable logic parts and EGL can convert these parts into executable Java or COBOL programs. Developers specify EGL logic by using EGL statements in EGL functions. Generatable part types are classified by their number of entry points and the manner of their invocation. ExternalType part belongs to prototype logic part and contains no logic of their own; instead it provides information needed to invoke methods associated with Java classes and objects [7].

In this paper, we will build a recommender system by ExternalType part calling Java classes. An ExternalType part provides EGL mapping to an external language element and an ExternalType part consists of variable declarations, function prototypes and constructor prototypes. This is similar to the mapping that an Interface part provides for Service functions, but generalized to include fields and constructors. The only external element that the part supports is the Java object. In the case of ExternalType, EGL does not have direct access to the code; we must provide certain prototypes so that EGL can perform the necessary type checking [7].

III. Recommender System Related Works

Personalized recommendation is an important part in user behavior analysis. In simple term, it is a process of searching the resource which the user might interest in. The technology has developed for decades [8]. At present, Collaborative Filtering is a relatively mature and popular recommendation algorithm. In collaborative filtering, an item is considered as a black box-we don’t look at its content and user interactions (such as rating, saving, purchasing) with the item are used to recommend an item of interest to the user [9]. More formally, given a dataset of user actions also called the user-item dataset, we will recommend items to a current user.

The main idea of Collaborative Filtering is to exploit information about the past behavior or opinions of an existing user community for predicting which items the current user will most probably like or be interested in. Collaborative Filtering approaches take a matrix of given user-item ratings as the only input and typically produce the following types of output: a numerical prediction indicating to what degree the current user will like or dislike a certain item and a list of N recommended items. Such a top-N list should not contain items that the current user has already rated [10].

Collaborative Filtering recommendation algorithm is divided into two parts: the user-based and the item-based collaborative filtering. The user-based collaborative filtering is one most used recommendation algorithm and the recommender system of e-learning system in this paper will use it to implement personalized items recommendation. It assumes if users had similar ratings about an item in the past, they will have similar ratings about the item in the future. That is, users’ preference is stable and constant over time.

The main idea is simply as follows: first given a user-item rating dataset and a current user as input and calculate the similarity between users; second use the similarity matrix made up of the above similarities to find similar users that had similar preferences to those of the current user, that is called nearest neighbors; third for every item that the current user has not yet seen, the prediction ratings will be calculated based on the ratings for the items made by the similar users; finally sort the prediction ratings and the top-N items will be recommended to the current user. Fig. 1 shows the recommender process:

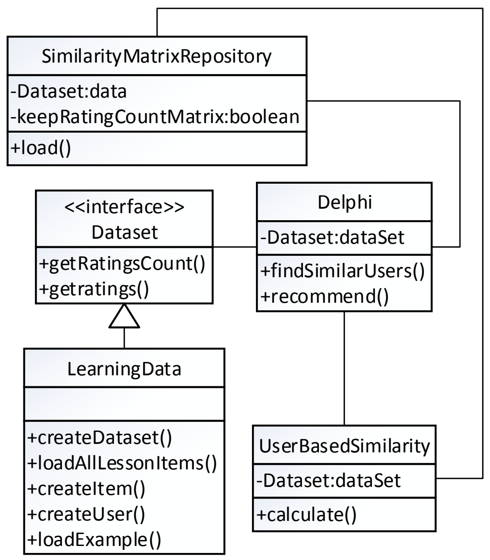

Fig. 1. The Class Diagram of user-based CF recommendation.

As is shown in Fig. 1, the *LearningData* class implements the *Dataset* interface and creates the dataset of lessons and students. At the moment, the current student does not know what lessons are appropriate to. Assume that we use a 1-to-5 scale to rate items in this paper, and rating “Lesson1” with a “5” means that the student strongly like this lesson and rating “Lesson1” with a “1”means that the student dislike it. The recommendation engine is the core part of recommender system to implement personalization. The *Delphi* class in Fig. 1 plays the role of recommendation engine and contains two vital methods: *findSimilarUsers()* and *recommend()*.

Recommendation engine aims to show items of interest to a user by users' interactions such as rating items here and in essence is matching the engine that take into account the context of where the items are being shown and to whom they're being shown[9].

The *findSimilarUsers(User user, int topN)* method in *Delphi* class is responsible for finding the N nearest neighbors. Usually N is five because the similarity will has bad effect on the accuracy of recommendation if N is too large or too small. The *Delphi* class must call *calculate()* method of *UserBasedSimilarity* class by calling *load()* method to work. The *calculate()* method contains similarity calculation method based on users. There are several similarity calculation methods, for instance the cosine similarity measure, Jacobi measure and Euclidean distance. Cosine similarity is established as the standard metric; as it has been shown that it produces the most accurate results [10]. The formula is as follows and $r_{u,a}$ is the rating given by user a to an item, $r_{u,b}$ is the rating given by user b to the same item.

$$\cos sim(a,b) = \frac{\sum_{u\in U}(r_{u,a} * r_{u,b})}{\sqrt{(\sum_{u\in U} r_{u,a})^2}\sqrt{(\sum_{u\in U} r_{u,b})^2}} \tag{1}$$

The value of defining similarity by Jacobi measure is a ratio as is shown below:

$$similarityValues[u][v] = \frac{agreementCount}{totalCount} \tag{2}$$

The $agreementCount$ represents the intersection of two users' ratings and $totalCount$ is on behalf of the union of two users' ratings, that is total number of ratings of the two users. The $similarityValues[u][v]$ stores the values of similarity based on users and it is a two-dimension double array named *SimilarityMatrix*. Therefore, the range of the ratio represents similarity is from 0 to 1. The ratio "0" represents that there is no similarity between two users and "1" shows that the two users' reference is completely consistent. The larger the ratio is, the higher the similarity between users is. In addition to the above similarity computing method, we can also use Euclidean distance to define the similarity. For all the users, calculate the deference of the ratings of each item between the two users and then Square, add them up to return the result value. Finally, extract square root of the above result value. The value obtained by the process is called Euclidean distance as is shown in formula (3).

$$d = \sqrt{\sum_{u\in U}(r_{u,a} - r_{u,b})^2} \tag{3}$$

But the Euclidean distance does not directly measure the similarity among users. Because usually the concept of similarity is contrary to the concept of distance, that is, the smaller the distance is, the higher the similarity is. We need reverse the relation. At the same time, we should also consider the accuracy of metric method of similarity. So we can define such a formular as is shown below:

$$sim(a,b) = 1 - \frac{\sqrt{\sum_{u\in U}(r_{u,a} - r_{u,b})^2}}{commonItems} \tag{4}$$

The ratio of the Euclidean distance and items that two users a and b rated jointly is a function curve of hyperbolic tangent and then we use 1 to subtract the ratio so that we can get a similarity value that its' range is from 0 to 1. This is an effective metric method of similarity. After the nearest neighbors are found, the *recommend()* method of *Delphi* recommender engine will do work according to the prediction rating of non-rated items based on the current user. The prediction rating formula is shown below:

$$predictedRating = \frac{\sum(sim(a,b) * itemrating)}{\sum sim(a,b)} \tag{5}$$

where, the prediction rating is the ratio of similar users' rating weighted sum and similarity sum. Then add up all similar users' prediction rating and sort them according to the values of sum and the top N items will be the recommendation items for the current user. The larger the prediction rating sum of an item rated by all similar users is, the more effective recommendation results the current user will get. Correspondingly the scale of the sum will not be more than "5". The success of collaborative filtering relies on the availability of a sufficiently large set of quality preference ratings provided by users. Accordingly, finding users with similar preferences is difficult if the user-item rating matrix is very sparse (few preference ratings), it will cause the sparse problem for the Collaborative Filtering method. In addition, the Collaborative Filtering method may suffer from the new item problem, in which there is no rating record on new items by which to derive the prediction [11].

IV. Implementation of the Personalized e-Learning System

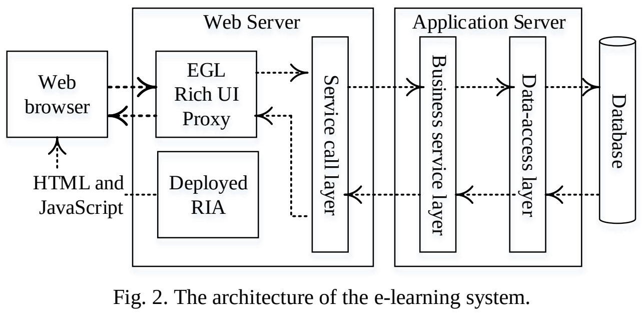

Fig. 2. The architecture of the e-learning system.

A. Application Architecture

As Fig. 2 showed, a kind of hierarchical and extensible e-learning system is proposed in this paper. After a Web server transmits the RUI (Rich User Interface) application to the user's browser, subsequent interaction with the server occurs only if the browser-based code accesses a service, which is a unit of logic that is more-or-less independent of any other unit of logic. The RUI application can access any number of discrete services. You automatically deploy the RUI application with the EGL RUI Proxy, which is EGL runtime code that handles the communication between the RUI application and the accessed services.

The architecture proposed by this paper makes full use of the characteristic of EGL technology and obtains further depuration and encapsulation, which enable the framework more suitable for the design and development of specific e-learning system [12]. The distinction among the tiers gives you a way to think about the different kinds of processing that occurs at run time. A tier is logical in the sense that some or all can be on the same machine. A service might be half a world away from the user. The RUI application can access any number of discrete services, and each might do a simple task such as provide details on a property. However, enterprise development often involves access of service-oriented applications. Each of these applications is composed of services that work as a unit to fulfill a specific and complex purpose.

In the development of application server layer, the RUI design of e-learning system is very important and the basic design idea is to refresh RUI widget as a unit. That is, the whole page of Handler part is divided into several widgets and each widget varies independently. When retrieving data by calling back-end service, EDT need to refresh the front page and the grain size of refreshed widgets should be as small as possible, so that it can reduce the throughput and response time. In order to realize the design, a global access control point of the refresh widgets is needed. EGL access service asynchronously, and each calling must construct a callback function instance which is responsible for refresh the user interface after returning the calling results. The creation of pages is simplified greatly by the use of widgets that are available within the page visual editor.

The business logic, which in most cases involves accessing data stores such as relational databases, can be found in the EGL Libraries, Programs or ExternalType part. EGL Programs part can be used to package up business logic with a single entry point. EGL ExternalType part is responsible for calling external language elements such as Java language and contains the functions and constructors of Java classes without the logic and relation of Java classes.

In the Data-access layer, the basic idea of a relational database is that data is stored in persistent tables. Access to relational database is by way of SQL in EGL Service part. Services can include new logic and can expose the data returned from other services and from called programs. EGL offers end-to-end processing: developers can write user interface, service logic, if necessary, new backend programs.

The application architecture with EGL is simple, easy to use and there is the same data structure in front-end and back-end development, so that developers can achieve the development of an application with EGL without mastering two languages.

B. *ExternalType Call Java*

EGL ExternalType can conveniently integrate Java language. In the EGL project of e-learning system application, the Java classes of personalized recommender system part are stored in the folder named “src”. All the Java classes and interface are created in the java package. The ExternalType part should be created in the client package of EGLSource folder. An ExternalType part consists of variable declarations, function prototypes and constructor prototypes. Take the *Delphi* class for example, the corresponding Externaltype shows below:

```
package client;
import eglx.java.*;
externalType Delphi type JavaObject
   {PackageName = "lessonrecommender"}
   constructor(arg0 Dataset? in);
   function findSimilarUsers(arg0 User? in)
   returns(SimilarUser[]?);
   function findSimilarUsers(arg0 User? in, arg1 int in)
   returns(SimilarUser[]?);
   private function findFriendsBasedOnUserSimilarity
(arg0 User? in) returns(List?);
   function getDataset()
   returns(Dataset?);
   function predictRating(arg0 User? in, arg1 Item? in)
   returns(float);
   function recommend(arg0 User? in, arg1 int in)
   returns(List?);
   // more function prototypes go here
end
```

As with other parts, an ExternalType definition reserves no storage, so we must declare a variable based on the part, optionally using the EGL new statement to be able to use its variables and functions and an ExternalType part can represent a Java class directly [7]. We can use the variable just like using the name of an EGL library. The variable provides access to public variables in the external language element. In addition, the name of the ExternalType part can be the same as the name of Java class under the premise of noting the name of the package that holds the Java class as the statement *{PackageName = "lessonrecommender"}* shows, so Java packages and EGL packages can work in much the same way. The reserved word “type” follows the name of ExternalType and specifies the type of external language element, here it is Java language.

The ExternalType part also requires a constructor, and the code *constructor (arg0 Dataset? In)* is the constructor of *Delphi* ExternalType part and its’ parameters is *Dataset*. Similarly, *Dataset* should also be an ExternalType part. In EGL terms, a constructor is a function that runs when a new variable based on the part is created. Constructors are not commonly used in EGL.

Within the ExternalType part, we should create function prototypes that represent the methods in the Java class that we need to use so that we can achieve the development of the functionality of recommender system. The function prototypes list the function or method name, its arguments, and its return value using EGL syntax, but no internal logic [6]. A prototype links a function in the ExternalType part to a method in the Java class. Also, we must be careful to match Java types to compatible EGL types, for example, double type in Java must be transformed into float type in EGL and short type in Java should be transformed into smallint type.

C. Invoke ExternalType Functions in EGL Program

An EGL Program part is a generatable logic part with a single point of entry. Each Program part contains a function named *main()*, which represents the logic that runs at program start up. A program can include other functions and can access functions that are outside of the program. The function *main()* can invoke those other functions, and any function can give control to other programs [7].

Like other EGL parts, the Program part uses stereotypes to specialize its code for particular user interfaces. The BasicProgram is the only stereotype that is part of the core EGL package. A BasicProgram can access databases or files, perform calculations, and use most of the EGL statements [7]. However, a BasicProgram cannot communicate with the user through a browser and cannot generate JavaScript language. There may be some limit in the development of Web 2.0 application. But each of the UI technologies offers additional stereotypes. We can create an associated project to call Java, and then relate it to the main project of e-learning system.

In the function *main()* of an EGL program part, we should create a variable based on the ExternalType using EGL new statement and use it in much the same way as a library name, appending the variable name to the name of the method using dot syntax. The below code shows the above process and creates a dataset of students and lessons named "*ds*".

```
myLearningData LearningData = new LearningData();
ds BaseDataset = myLearningData.createDataset();
```

myLearningData is the declared variable name of the ExternalType named *LearningData*. The code *new LearningData()* is actually a function call to the *constructor(arg0 Dataset? in)* function prototype in the part definition, which in turn refers to the constructor of the Java class. The statement *myLearningData.createDataset()* is used to call the function *createDataset()* of ExternalType *LearningData*. Then we can use the recommender engine to recommend lessons for the current student.

```
delphiU Delphi = new Delphi(ds);
delphiU.setVerbose(true);
stu StudentUser = ds.pickUser("Babis");
delphiU.findSimilarUsers(stu);
delphiU.recommend(stu);
```

By the function *pickUser()*, we can select a random student such as Babis as the current student and call the function *findSimilarUsers()* of recommender engine *Delphi* to find the nearest neighbors for the student. Finally, the core function of recommender engine *Delphi* named *recommend()* will provide the effective recommendation items.

The running result of finding nearest neighbors shows below:

TABLE I: TOP FRIENDS FOR USER BABIS

Name	Similarity
Carl	0.857143
Aurora	0.714286
Alexandra	0.666667
Athena	0.625000
Charlie	0.571429

Through the values of similarity we can see that Carl' preferences are more similar to Babis' and Aurora is the second. Then the recommendation items for Babis are shown below:

TABLE II: RECOMMENDATIONS FOR USER BABIS

Item	Predicted rating
Precision Mechanical and Manufacturing Engineering	4.285700
Basic of Electronic Technology	3.333300
The principle of Automatic Control	1.428600

The predicted ratings of top two items are 4.28 and 3.33, and they are much closer to 5.00 than the third item. This indicates that the current student Babis should firstly select Precision Mechanical and Manufacturing Engineering or Basic of Electronic Technology to study and the two lessons are more suitable for him than the third lesson.

V. CONCLUSION

In the personalized recommender system, the core feature of personalized information is user-centered and to generate the recommendation items which are available and reflect the preferences of individuals. The recommender system of e-learning system in this paper is implemented based on EGL. Fig. 3 shows the aggregate average response time (AART) of the system of the application architecture based on open source framework – Struts, Hibernate, Spring (SHS), Java EE and EGL. The testing scenario is information query in the program of client-side.

Analysis of AART curve in Fig. 3, compared with the solution of SHS, the system response time of the solution of Java EE increases in accordance with the linear way until it reaches about 100 users. After that time, the increase of the curve becomes more intense. By analyzing the change trend of the two curves, we can draw the conclusion: the maximum number of users of the solution of Java EE presents is 100. Because EGL can be executed after generating Java code, the changes of the curve of the solution of EGL are similar to Java EE's and only its speed is slightly slow.

Compared with the solution of Java EE, the solution of EGL has some great advantages. The application architecture based on EGL implements the loose coupling between business logic and access control, that is, choices related to a user interface, and choices related to the data in persistent, are handled separately. In the development of EGL Rich UI application, the MVC pattern is often used to realize above-mentioned features. The separation of Model and View also allows for a division of labor. This division lets developers fulfill a task appropriate to their profession and lets different tasks proceed in parallel. Also, EGL is general because the technologies are so varied, but we will follow up by nothing how the separation applies to EGL Rich UI. Therefore, the primary benefit of the application architecture based on EGL is simple, easy to use and across languages, frameworks and runtime platforms; it not only can avoid the repeated writing the similar logic of each domain module, also is conducive to the robustness, maintainability of the system, and flexibility to the changes of whole business needs.

The solution of EGL can significantly reduce the cost and complexity of e-learning system and improve the e-learning system interactive experience and real time requirements.

The e-learning system discussed in this paper will have an important guiding significance for the application and development of the e-learning system in the network education. At the same time, how to deploy and test EGL application is the next issue we should concern about.

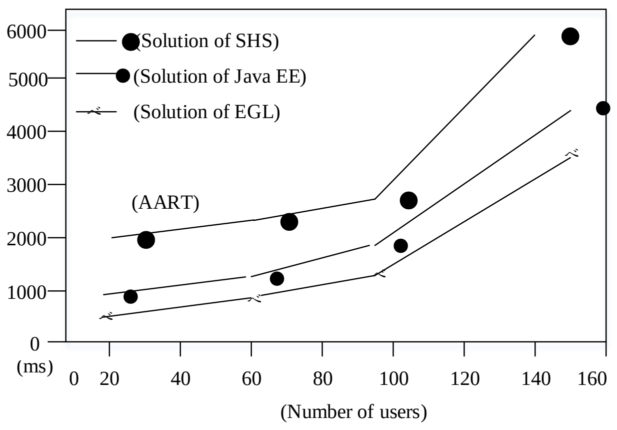

Fig. 3. AART curve.

ACKNOWLEDGMENT

This work was supported by the Science and Technology Project of Education Department of Liaoning Province, China (Research of e-Learning System of small and medium-size Enterprises Based on SaaS, No.L2013417).

REFERENCES

[1] L. Ren, "The Key Technology Research of Recommender System," East China Normal University, 2012.

[2] R. Francesco, R. Lior, B. Shapira, and P. B. Kantor, *Recommender System Handbook*, pp.145-147, Springer, 2011.

[3] B. Song and M. Y. Li, "An e-learning system based on GWT and berkeley DB," *Lecture Notes in Computer Science 7332*, vol. 2, pp. 26-32, 2012.

[4] W. Fang and Y. Sun, "Research and application of J2EE's data persistence layer," *Computer Technology and Development*, vol. 17, no. 2, pp. 68-91, 2007.

[5] B. Song and Y. Zhang, "Implementation on Network Teaching System Based on Java EE Architecture," in *Proc. IEEE Second International Conference on Information Technology and Computer Science*, vol. 1, Kiev, pp. 354-357, 2010.

[6] IBM Corporation, *EGL Programmer's Guide Version7 Release 00*, USA, 2007.

[7] IBM Corporation, *EGL Language Reference Version7 Release 01*, USA, 2007.

[8] J. Xiao and L. He, "An expandable recommendation system on IPTV," *Lecture Notes in Computer Science 7332*, vol. 2, pp. 33-40, 2012.

[9] S. Alag, *Collective intelligence in action*, pp. 349-350, Manning Publications, 2009.

[10] J. Dietmar, Z. Markus, F. Alexander, and F. Gerhard, *Recommender system an introduction*, pp. 13-19, Cambridge University Press, 2011.

[11] Q. Liu, Y. Gao, and Z. Peng, "A novel collaborative filtering algorithm based on social network," *Lecture Notes in Computer Science 7332*, vol. 2, pp. 164-174, 2012.

[12] B. Margolis, *Enterprise Web 2.0 with EGL*, MC Press: USA, pp. 131-141, 2009.

Jie Gao is currently studying for a master's degree in Software College of the Shenyang Normal University (SYNU). She obtained the bachelor degree from Yantai Nanshan University in 2012. Her main topics of interest include software engineering, e-learning, collective intelligence and personalized information process.

Bo Song is a full professor of the Software College of the Shenyang Normal University. He obtained the M.S. degree from the University of Fukuoka Education, Japan in 1999. His main topics of interest are software engineering, e-learning, collective intelligence and personalized information process.