The Architecture of a Learner Adviser Service

Dumitru Dan Burdescu[1], Marian Cristian Mihăescu[1] and Costel Marian Ionaşcu[2]

[1]Software Engineering Department, University of Craiova, Romania
{burdescu, mihaescu}@software.ucv.ro
[2]Analysis, Statistics and Mathematics Department, University of Craiova, Romania
icostelm@yahoo.com

Abstract. One of the most important challenges faced by institutions that deploy e-Learning activities is to prove beneficiaries that the learning process is effective. This paper proposes a structure for a Service Oriented Architecture (SOA) in which a Learner Adviser Service (LAS) will run. The proposed architecture enables different e-Learning platforms to access the service through published and discoverable interfaces. The LAS will provide feedback for each learner according with the setup that has been done between the e-Learning platform and LAS. The feedback refers to actions that are recommended to be performed by the learner. LAS uses machine learning algorithms for classifying learners according with their performed activities. The ultimate goal of LAS is to provide an overall activity measurement for the student's activity in such a way to increase the trust into the effectiveness of the e-Learning platform.

Keywords. Service Oriented Architecture, e-Learning, Recommender system.

1 Introduction

Using service type architecture has the advantage of providing the ability to register, discover, and use services, where the architecture is dynamic in nature.

A typical Service-Oriented Architecture is presented in Figure 1. One can see three fundamental aspects of such architecture:

a) Advertising. The Service Provider makes the service available to the Service Broker.

b) Discovery. The Service Consumer finds a specific Service using the Service Broker.

c) Interaction. The Service Consumer and the Service Provider interact.

In an e-Learning context, the service consumer is represented by the e-Learning platform itself, while the service provider is represented by LAS. Once a service is advertised and discovered, any interaction with it is controlled by the service level agreement (SLA) that is going to define the entities that are allowed or denied access to the service, as well as the interval time the access is allowed or denied.

The lifetime of LAS can be described by the three following steps: creation, advertising and discovery. Each of these steps is represented in LAS by a set of interfaces. There will be explained these different steps and demonstrate them through a prototype LAS.

Burdescu D., Mihaescu M. and IonaÅ§cu C.
The Architecture of a Learner Adviser Service.
DOI: 10.5220/0004463200570070
In *Proceedings of the 3rd International Workshop on Enterprise Systems and Technology (I-WEST 2009)*, pages 57-70
ISBN: 978-989-674-015-3

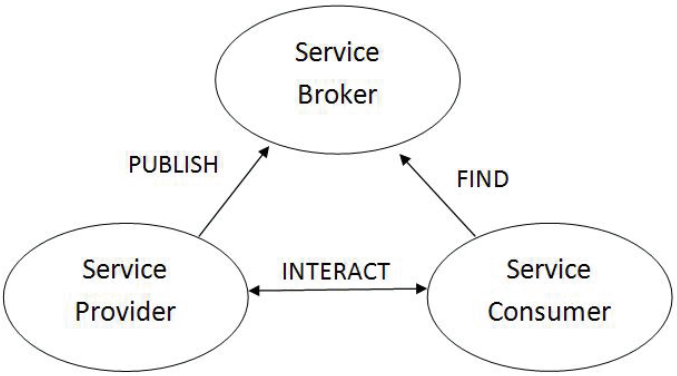

Fig. 1. The Service-Oriented Architecture.

As seen before, the different application areas have different requirements to the data management. Workflows need a reliable place for the storage of temporal or permanent processing results. The usage of content sharing applications needs sophisticated querying functions and short access times. Applications based on resource sharing require standardized reliable database access. Finally, collaborative applications require concurrency control.

In the service oriented applications, the same classical requirements that apply to databases are still valid:

Durability: Data need to be stored for longer time. In web application, data have to be available even if storage peers disappear.

Consistency: Data have to be always consistent. This is challenging in web application as data can be changed in every storage peer and changes have to be propagated to other peers.

Reliability: The reliability of web application data stores is accomplished by the distributed storage of data. Hence the reliability of web application storage is related to the durability property.

Concurrency: From the architecture of web application system a high level of concurrent operations on data is given. Changes can be done on several peers in parallel. Hence updates have to be done in a more controlled way.

Scalability: The scalability is base property of web applications.

The LAS employs machine learning algorithms in order to create a knowledge management infrastructure that has as main goal building trust for the obtained advice. The input for the business logic of LAS is represented by learner's activities and by goals. Learner's activities represent the data used for creating learner's model. The goals represent the criteria that need to be optimized in order to obtain quality advice.

Advice obtained by LAS regard the level of fulfilling proposed goals. This is an objective measure of the quality of the evaluation environment. On the other hand, the recommendations represent advice for course managers and learners. The aim of advice is to increase the quality of the e-Learning process. The procedure consists of several steps. Firstly, the platform has to produce enough data regarding the learner's performed activities such that a learner's model of good quality is obtained. At this

step there are also set up goals. Course managers set goals regarding their course and learners set up their own goals. This step is called SETUP and is considered to be the most important one since next steps heavily rely on it.

After the model has been obtained the next step is to obtain advice. The advice is supposed to be strictly followed by course managers. The period in which course managers carry out the recommendations is called EEI (Evaluation Environment Improvement). The activities performed by learners in this period will not be taken into consideration regarding in the learner's model or recommendations by the LAS. After the EEI period ends a new dataset of learner's performed actions is recorded. This dataset is used for rebuilding the learner's model and reevaluation of initially set goals. This step is called EER (Evaluation Environment Reevaluation).

Regarding the advice for learners, the e-Learning platform has implemented means of keeping track of advice that has been given to learners and the way the advice were followed. This is accomplished also in EER step. The LAS provides at this step conclusions regarding the quality of recommendations by evaluating whether or not the learners were helped to reach their goals or not.

This three step process may have as many iterations as needed. Each reevaluation step compares a challenger learner's model with initial model in terms of classification accuracy. The model with best accuracy will be further used for offering advice to learners. The challenger model is based also on newly recorded data from the time old model has been obtained. It is a primary concern to continuously improve the learner's model in terms of classification accuracy. This is the basis for obtaining valuable advice for learners and course managers.

2 Related Work

Several service definitions exist in the literature, e.g., [3][4]. In general, services provide one or more functionalities to consumers. Consumers can be anybody that needs a specific functionality. They could be client applications requesting some external support, or other services as well. The client can be of any system, in or case an e-Learning application.

One important class of example for services are Web services. Web services use the available infrastructure of the Internet as the communication medium. They are built on a wide range of standards or proposed standards for service description and high-level communication protocols. All data formats use the XML [5] encoding. Services are described by WSDL [7] and published with the help of UDDI [6].

The individual service calls use SOAP [8]. For the implementation of distributed workflows BPEL [9] has been proposed as a new standard. With the introduction of a transaction protocol [10] also some work on the reliability of Web service architectures has been done. The strict focus on open Internet standards is an important difference to other approaches of distributed computing like Corba [11], or DCE [12]. Since only the strict interface details are published, Web services are neutral to the programming language, programming model and the underlying operating system.

The business logic of LAS makes intensive use of machine learning talgorithms implemented in a knowledge management infrastructure. From this perspective LAS benefits from knowledge management concepts and technologies.

Knowledge is considered to be “the information needed to make business decisions” [13], and so knowledge management is the “essential ingredient of success” for 95 per cent of CEOs [13].

3 Design and Implementation of LAS

3.1 Creation, Advertising and Discovery

A service is defined by its interface, i.e. the list of methods it provides. For example, the interface and the implementation of an adviser service providing basic functionalities to find and classify a learner. It is important at this level to notice there is no information on how the service is going to be implemented. We will see in the following sections how this abstract LAS is going to be implemented by using for example the Jini library.

The instantiation of the service is done through a call to a meta factory, which first instantiates a service factory for the used implementation, and asks this factory to return a new instance of the service.

Once created, a service can be advertised on a specific domain through the advertizing manager service. The service is advertised with a SLA that defines the access policy that will be used to enforce interaction with the service. The same service can be advertised in different organizations (e.g. other e-Learning platforms) with different SLA’s. This gives a flexible mechanism to control how different organizations may access the service, by allowing advertising the service capabilities as required.

The advertising of a service is done through a XML document that defines the SLA’s of the service for all the virtual organizations where the service is to be made available.

By connecting to a virtual organization, a service consumer can query a service and interact with it once discovered. LAS provides different types of query such as interface matching that allow to listen to all services of a specific interface, or service data matching that allow to query services based on the value of their service data elements. Discovering a service is accomplished in three steps: (1) Instantiate a discovery manager, (2) Instantiate a discovery query - the instantiation mechanism is based on the interface of the service will listen, (3) Register a listener - for every new service matching the query, the *service-Published()* method will be called with the service as a parameter. Similarly, the *service-Unpublished()* method will be called for each service disappearing from the virtual organization.

Any interaction with the service is controlled by an external entity; it first authenticates the service consumer through its certificate and authorizes it against the policy of the service it wishes to access.

Jini network technology [1] is an open architecture that enables developers to build adaptive networks that are scalable, evolvable and flexible as typically required in

dynamic computing environments. The first version of the LAS was directly implemented on top of the Jini API [2].

When using Jini, the following classes are automatically generated for a service named LAS.

– LAS_ServiceJiniNoAbstract.java extends the implementation of the service LAS to provide an implementation for all the basic LAS/Jini mechanisms.

– LAS_ServiceJiniStub.java is the main Jini interface extending the interface java.rmi.Remote. It acts as a proxy for LAS service, and defines exactly the same methods.

– LAS_ServiceJiniStubImpl.java is the implementation of the interface LAS_ServiceJiniStub. It uses a reference to LAS_ServiceJiniNoAbstract to redirect all the method calls on the service.

– LAS_ServiceJini.java implements the interface LAS by using a reference to LAS_ServiceJiniStub to redirect an LAS service's method call as a Jini service's method call.

Step 1: Creation. This step creates an object of the class LAS_ServiceJini and initializes it with the corresponding stub, i.e. an instance of the class LAS_ServiceJiniStub-Impl.

Step 2: Advertising. The object LAS_ServiceJiniStubImpl – hold by the LAS service created in the previous step – extends indirectly the interface java.rmi.Remote, it can therefore be made available in a Jini lookup service.

Step 3: Discovery. The object returned from the Jini lookup service is a LAS_ServiceJiniStubImpl. It is going to be wrapped in an instance of the class LAS_ServiceJini before being returned to the listener. We obtain here a similar object to the one obtained when creating the service.

Step 4: Invocation. Any method call is done on an instance of the class LAS_ServiceJini and is finally redirected on an instance of the class LAS_ServiceImpl.

Figure 2 shows an interaction diagram of these different classes and interfaces.

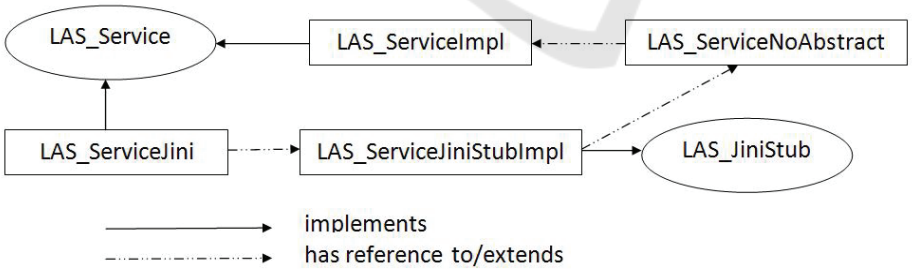

Fig. 2. Jini Implementation of an LAS Service.

3.2 Advantages/Disadvantages

The functionalities provided by the SOA of LAS and the Jini library are basically the same. It was therefore very easy to implement the SOA on top of Jini without tying up LAS to Jini and get an implementation-independent SOA. The Jini implementation is very scalable; experiments of testing the performance of Jini when increasing the number of Jini services, demonstrate a good result in the performance when discovering and accessing the Jini services. The potential problems when using Jini lie in security and in the connection of services across firewalls.

3.3 System Prototype Design

Based on key requirements of e-learning systems, we can prototype the LAS. This service may respond to queries from three types of users: learners, instructors and administrators. Figure 3 presents a LAS, and two e-Learning systems, Uni-1 and Uni-2, which benefit from the same services, implemented in LAS.

Users (learners, professors, administrators) from different universities (e-Learning platforms) can benefit from using the LAS. Each university is required to be registered with LAS and a setup procedure needs to be accomplished. This procedure regards the way in which the universities will provide activity data to LAS. Once this setup in accomplished, the universities will be able to query LAS using the specified interface.

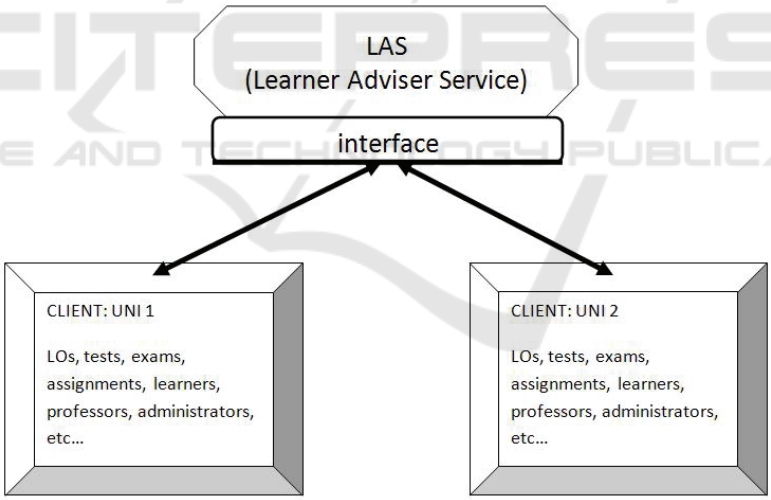

Fig. 3. General architecture design.

From logical point of view, the architecture will have the following layers:

- LAS – Learner Adviser Service itself. This layer is represented by the business logic implementation of the provided services;

- Client Representation level – at this level, within LAS, there are stored the data regarding a registered client (e.g. an e-Learning system). These data regard the locality of client, the format in which data is sent, the credentials of the client.
- LAS interface – this level regards the way in which LAS may be queried by clients;
- Access control layer – Authenticates and authoresses users such that there are determined what resources a user is allowed access. Administration refers to the ability to specify the way LAS and client interact.

The Client level – this level is represented by the e-Learning platform itself.

3.4 Framework Implementation

Figure 4 presents the implementation framework of LAS and an e-Learning system. It focuses on the implementing flows between different layers.

From the container (e.g. an Internet browser), in which the client contents are presented by the forms of HTML, JSP (Java Server Pages) or XML, the user (learner, professor, administrator) can submit their query through HTTP requests to the Servlet Container and receives HTTP responses. The query goes through the Database Connection Layer (e.g. JDBC (Java Data Base Connection) API) to get information from the database.

Regarding LAS, the request goes from Servlet container to the Service agent which parses the request streams and transmits the data to LAS. Once request is fulfilled the same Servlet container will get back the response.

The procedure may be enhanced by doing a look up in a UDDI registry, the Web Service Agent Layer can search and locate the LAS. The location of WSDL binding information will be sent back as a SOAP message. The binding information of our own sharable learning services is also published here. After the service Agent Layer gets the binding information, it can directly invoke the learning service by passing the essential data indicated by the WSDL file in a SOAP message over the Internet or Intranet. And the learning service could be J2EEbased or other platform-based residing on any platform.

Access control is the process to identify client applications (e.g. e-Learning platforms) that need to access LAS. Once a client has been authenticated, authorization determines what resources a client is allowed to access. Administration refers to the ability to modify the way in which the client interact with LAS.

Service Requestor is a component that needs to be integrated within the e-Learning application that wants to receive a service from LAS. It does not know where this other application is or how to locate it, so it turns to the Service Registry and requests a find operation. The Service Provider publishes this service as a Service Proxy in the Service Registry. Service Proxy is a java proxy that helps the system to communicate with web service.

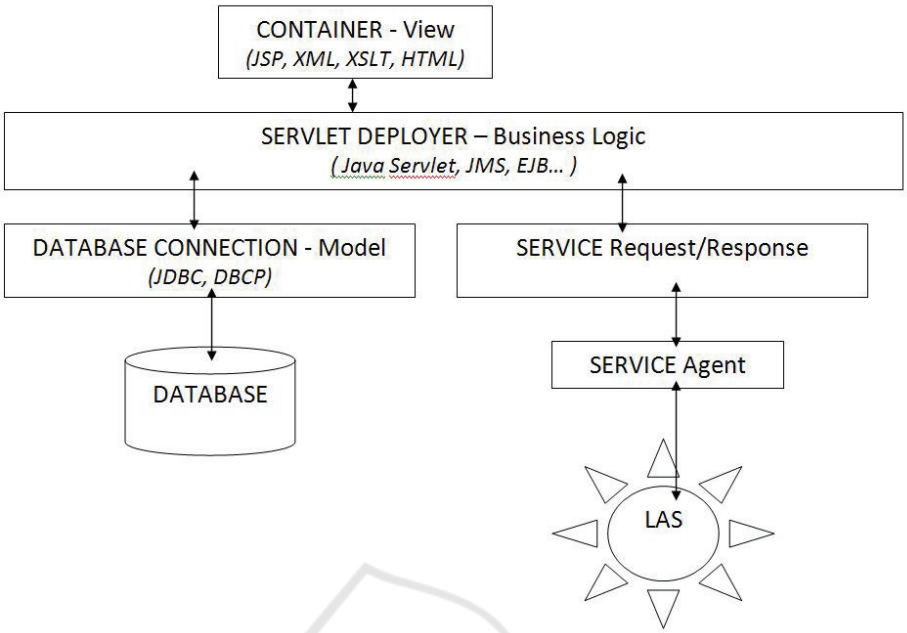

Fig. 4. Implementation framework of LAS and e-Learning system.

3.5 Business Logic of LAS

The whole process is conducted following the steps of target modeling presented in figure 5 [14].

Defining the goal represents the first step. Our goal is to create a model of analysis for Tesys e-Learning platform that is to be used for optimizing the criteria specified by learners and course manager goals. Setting up the goals is accomplished by formally defining the criteria that is to be evaluated and optimized. Selection and preparation of data are the next steps. Here, we have to determine the necessary data that will enter the modeling process. The preparation gets that data and puts it into a form ready for processing of the model. Since the processing is done using machine-learning algorithms implemented in Weka workbench [15], the output of preparation step is in the form of an *arff* file. Under these circumstances, we have developed an offline Java application that queries the platform's database and crates the input data file called *activity.arff*. This process is automated and is driven by a property file in which there is specified what data will lay in *activity.arff* file.

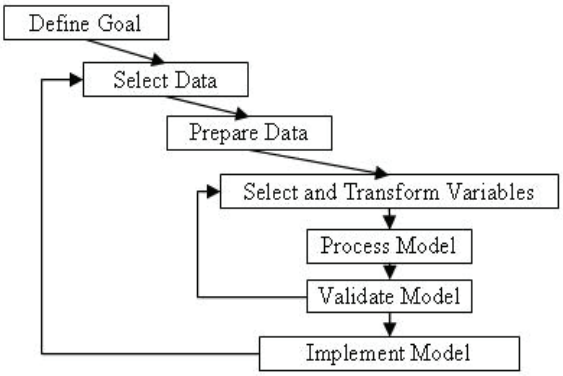

Fig. 5. Steps for target modeling.

For a learner in our platform we may have a very large number of attributes. Still, in our procedure we used only attributes related to testing procedures: the number of taken tests, average time spent on tests and average results obtained. Here is how the *arff* file looks like:

```
@relation activity
@attribute noOfTests {1, 2, 3, 4, 5}
@attribute avgTimeForTesing {1, 2, 3, 4, 5}
@attribute avgResultsOnTests {1, 2, 3, 4, 5}
@attribute avgFinalResults {1, 2, 3, 4, 5}

@data
3,3,4,5
4,4,5,4
……
```

As it can be seen from the definition of the attributes each of them has a set of nominal values from which only one may be assigned. The values of the attributes are computed for each of the 650 learners and are set in the *@data* section of the file. For example, the first line says that the learner took an average number of tests, spent an average amount of time for testing, and obtained good results at testing and very good results at final examinations.

Now, since we have prepared the data we start analyzing it. Choosing between two learning algorithms given a single dataset is not a trivial task [16]. Firstly, we make sure the data is relevant. We test the "goodness" of data trying to build a decision tree like C4.5 [17] from data. A decision tree is a flow-like-chart tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes [18].

The basic algorithm for decision tree induction is a greedy algorithm that constructs the decision tree in a top-down recursive divide-and-conquer manner [18].

The computational cost of building the tree is O(mn log n) [15]. It is assumed that for n instances the depth of the tree is in order of log n, which means the tree is not degenerated into few long branches.

The information gain measure is used to select the test attribute at each node in the tree. We refer to such a measure an attribute selection measure or a measure of goodness of split. The algorithm computes the information gain of each attribute. The attribute with the highest information gain is chosen as the test attribute for the given set [18].

Consequently the data must be preprocessed to select a subset of attributes to use in learning. Learning schemes themselves try to select attributes appropriately and ignore irrelevant and redundant ones, but in practice their performance can frequently be improved by preselection. For example, experiments show that adding useless attributes causes the performance of learning schemes such as decision trees and rules, linear regression, instance-based learners, and clustering methods to deteriorate [18].

In the tree building stage, the most important step is the selection of the test attribute. Information gain measure is used to select the test attribute at each node in the tree. Such a measure is referred to as an attribute selection measure or a measure of the goodness of split. The attribute with the highest information gain (or greatest entropy reduction) is chosen as test attribute for the current node. This attribute minimizes the information need to classify the samples in the resulting partitions and reflects the least randomness or "impurity" in these partitions.

Finally, the cross-validation evaluation technique measures the correctly and incorrectly classified instances. We consider that if there are more than 80% of instances correctly classified than we have enough good data. The obtained model is further used for analyzing learner's goals and obtain recommendations. The aim of the LAS is to "guide" the learner on the correct path in the decision tree such that he reaches the desired class.

The main characteristic of the LAS is that it uses a machine learning algorithm for obtaining knowledge regarding learners. The e-Learning environment produces data regarding the activity of learners and passes this data to LAS. The LAS creates and maintains a learner's model based on data received from the e-Assessment tool. This architecture allows the usage of LAS along with any e-Learning platform as long as the data is in the accepted format.

The raw data is dumped by e-Learning platform in a log file *activity.log*. The log file, together with database relations represent the raw data available for the analysis process. Because we use Weka [19] the data is extracted and translated into a standard format called ARFF, for Attribute Relation File Format [20, 21]. This involves taking the physical log file and database relations and processing them through a series of steps to generate an ARFF dataset.

At this phase the most important decision regards the features selection for instances. There may be derived a large number of features that describe the activity of a student. Choosing the attributes is highly dependent on data that we have domain knowledge and experience. For our classification we choose three attributes: noOfTests– the number of taken tests, avgTimeForTesing – the average time spent for testing, vgResultsOnTests – average results obtained at testing and avgFinalResults-average of final results. For each registered student the values of these attributes are determined based on the raw data from the log files and database relations. Each student is referred to as an instance within classification process.

The values of attributes are computed for each instance through a custom developed off-line Java application. The outcome of running the application is in the

form of a file called activity.arff that will later be used as source file for Weka workbench [19].

The activity.arff file has a standard format which is composed of two sections. In the first one there is defined the name of the relation and the attributes. For each attribute there is defined the set of nominal values it may have.

At this point we may say we have obtained useful data that may be used for experimentation with machine learning schemes. The original dataset was divided into a training of 90% of instances and a test set of 10 % of instances. The model was constructed using four attributes: nLogings, nTests, avgTest and nSentMessages. The obtained decision tree represents the learner's model. This model is used as reference when analyzing learner's activity.

More detailed results regarding the obtained model are presented below.

```
=== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     activity
Instances:    375
Attributes:           3:  noOfTests,  avgResultsOnTests,  avgTimeForTesing,
avgFinalResults
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
noOfTests = 1: (25/2)
noOfTests= 2
|  avgResultsOnTests = 1 (20/1)
|  avgResultsOnTests = 2
|  |  avgTimeForTesing= 1
|  |  | avgFinalResults = 1 (10/3)
|  |  | avgFinalResults = 2 (27/10)
|  |  | avgFinalResults = 3 (7/2)
|  |  | avgFinalResults = 4 (5/1)
|  |  avgTimeForTesing= 2
|  |  |  avgFinalResults = 1 (20/6)
|  |  |  avgFinalResults = 2 (13/4)
|  |  avgTimeForTesing= 3 (7/1)
|  |  avgTimeForTesing= 4 (5/0)
|  avgResultsOnTests = 3 (113/6)
noOfTests = 3
|  avgResultsOnTests = 4 (11/3)
|  avgResultsOnTests = 5
|  |  avgTimeForTesing= 3 (17/4)
|  |  avgTimeForTesing= 4 (29/3)
|  |  avgTimeForTesing= 5 (35/4)
noOfTests = 4 (21/2)
noOfTests = 5 (12/2)
Number of Leaves  :     17
Size of the tree :      25
```

```
Time taken to build model: 0.13 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances   302       80.6    %
Incorrectly Classified Instances  73       19.4    %
```

The most important part is the results validation, which ensures that the model is valid and provides solid knowledge. The stratified cross-validation evaluation technique revealed that 302 (80.6 %) instances were correctly classified and 73 (19.4%) were incorrectly classified.

Whenever a learner performs specific actions he is classified by the decision tree and conclusions are obtained. These conclusions may be regarded by recommendations for learners having as final goal helping them in reaching educational objectives. This continuous monitoring and classification may have a big contribution to building a quality e-Learning environment.

Within the e-Learning environment there is specified a set of goals for learners and course managers. For learners the set of goals from which they may choose is:

- Minimization of the time in which a certain level of knowledge is reached. This is accomplished by specifying a desired grade.
- Obtaining for sure a certain grade. The learner has to specify the grade he aims for.

Course managers may choose from two goals:

- Having a normal distribution of grades at chapter level.
- Having a testing environment that ensures a minimum time in which learner reaches a knowledge level for passing the exam.

For these goals there were created two sets of recommendations. Learners may obtain one of the following recommendations:

- More study is necessary for chapter X.
- You may go to the next chapter.
- You need to take more tests at chapter X.

For course managers the set of recommendations is:

- At chapter X there are needed harder/easier questions.
- At chapter X there are to few/many questions.

For obtaining recommendations for course managers we have used the obtained model. Running the Decision Tree algorithm created 17 classes. For these classes, we compute the likelihood of a set of test data given the model. Weka measures goodness-of-fit by the logarithm of the likelihood, or log-likelihood: and the larger this quantity, the better the model fits the data. Instead of using a single test set, it is also possible to compute a cross validation estimate of the log-likelihood. For our instances, the value of the log-likelihood is -2.61092, which represents a promising result in the sense that instances (in our case learners) may be classified in four disjoint clusters based on their activity.

After the model has been created the recommendations towards course managers were made and the evaluation environment was altered accordingly. The recommendations and the behavior of learners (whether or not they followed recommendations) were logged for further analysis.

The behavior of learners has a very important role in obtaining challenger learner's models that at some point may replace the current one.

On the other hand, checking whether or not the learners followed the recommendations may lead to conclusions regarding the quality of recommendations and of currently employed learner's model.

4 Conclusions

The paper presents the design of a Service-Oriented Architecture for a LAS service that may be integrated with an e-Learning platform.

LAS have been designed as a complex Web service that may be used by e-learning systems. LAS web service, as a service provider, provides the response to the client' request.

The main purpose of LAS is to create a learner's model corresponding to a registered e-Learning platform. Once the model has been created, LAS may receive queries from the e-Learning platform regarding the actions that need to be performed by learners in order increase the trust into the effectiveness.

The software architecture of LAS uses only Java related technologies. That is why the system has an open architecture and uses open application interfaces to enable interaction and integration seamlessly between educational institutions and LAS. The architecture is able to take advantage of the open, dynamic nature of the web by supporting just -in-time application integration.

Our LAS produces advice for learners and course managers using different machine learning techniques on the activity data obtained from the platform. We use Weka workbench [19] as environment for running state-of-the-art machine learning algorithms and data preprocessing tools. We have developed a custom application that gets the activity data from the platform and transforms it into the specific file format used by Weka, called arff.

A decision tree learner is used for estimating whether or not the data may be used to obtain significant results. The outcome of decision tree validation is the percentage of correctly classified instances. We say that a value of over 80% in correct classified instances is a promise that we might finally obtain useful knowledge.

Clustering may also be used for estimating the classification capability evaluation environment. This is mainly be performed to obtain better recommendations for course managers.

We have tested this procedure on data obtained from the e-Learning platform on which 375 learners were enrolled and had activity for six month. The results are satisfactory and prove that the evaluation environment can be successfully used in an e-Learning process.

References

1. Jini Network Technology. http://www.sun.com/software/jini/
2. JiniTM Technology Starter Kit v2.0 API Documentation, http://java.sun.com/products/jini/2.0/doc/api/
3. D. Tidwell. Web Services - The Web's next revelotion. https://www6.software.ibm.com/ developerworks/education/wsbasics/wsbasic%s-ltr.pdf.

4. Web Services Architecture Requirements, October 2002. http://www.w3.org/TR/wsa-reqs
5. [3] Extensible Markup Language (XML) 1.0, October 2000. http://www.w3.org/TR/REC-xml
6. Universal Description, Discovery and Integration (UDDI), 2001. http://www.uddi.org/
7. Web Services Description Language (WSDL) 1.1, March 2001. http://www.w3.org/TR/wsdl
8. Simple Object Access Protocol (SOAP) 1.1, 200. http://www.w3.org/TR/SOAP/
9. [10]Business Process Execution Language for Web Services (BPEL), 2002. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
10. Web Services Transaction, August 2002. http://www-106.ibm.com/developerworks/library /ws-transpec/
11. Common Object Request Broker Architecture, 2002. http://www.omg.org/
12. OSF Distributed Computing Environment, 2002. http://www.opengroup.org/dce/
13. Manchester 1999Philip, Manchester, “Survey – Knowledge Management” Financial Times, 8 April,1999.
14. Olivia Parr Rud, Data Mining Cookbook – Modeling Data for Marketing, Risk, and Customer Relationship Management (Wiley Computer Publishing, 2001).
15. I. H. Witten, E. Frank, Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations (Morgan Kaufmann Publishers, 2000).
16. R. Agrawal and R. Srikant, Fast algorithms for mining association rules. Proc. of the 20th VLDB Conference, Santiago, Chile, 1994, pp. 487-499.
17. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
18. Jiawei Han, Micheline Kamber, Data Mining – Concepts and Techniques. Morgan Kaufmann Publishers, 2001.
19. Weka, www.cs.waikato.ac.nz/ml/weka
20. Garner S.R., Cunningham S.J., Holmes G., Nevill-Manning C.G. and Witten I.H., "Applying a Machine Learning Workbench: Experience with Agricultural Databases," Proc Machine Learning in Practice Workshop, Machine Learning Conference, Tahoe City, CA, USA, pp. 14-21., 1995.
21. Holmes, G., Donkin, A., and Witten, I.H., “Weka: a machine learning workbench.” Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, pp. 357- 361, 1994.