World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:4, 2009

Extending E-learning systems based on Clause-Rule model

Keisuke Nakamura, Kiyoshi Akama, and Hiroshi Mabuchi

Abstract—E-Learning systems are used by many learners and teachers. The developer is developing the e-Learning system. However, the developer cannot do system construction to satisfy all of users' demands. We discuss a method of constructing e-Learning systems where learners and teachers can design, try to use, and share extending system functions that they want to use; which may be nally added to the system by system managers.

Keywords—Clause-Rule-Model, database-access, e-Learning, Web-Application.

I. Introduction

In recent years, E-learning systems (LMS) are being used extensively by universities to provide online courses and also by companies to provide training to employees. In general, the functions of the LMS is constructed by a system developer, the teacher publishes the learning materials, the student learns by using the provided functions along with the materials, and the teacher grades the work submitted by the student. However, when an LMS is developed via this mechanism, its ability to provide specialized functions that a particular student and teacher may desire is very limited. The functions desired by users of a LMS are numerous and varied. Therefore it is impossible for a developer to implement all the functions a particular user may desire. Some studies with respect to LMS propose new frameworks which gives the users the capability to construct new functions. For example, in Moodle [1], a user is able to construct new functions called modules. However, the average user of the LMS still find it very difficult to construct the functions he or she desires. An e-Learning system is a Web application, and as such it is composed using WWW technology, network technology, and database technology. Therefore,in order to properly create new functions the average user would need to have related knowledge and know about the underlying technologies. It is not a free learning environment in which personal, new functions can be made by everyone. Additionally, the database of the LMS includes classified information with respect to students. It is essential that database security be taken into account even while freely allowing the average user to construct new functions that can access the database. All these challenges have to be taken into account by the developer when considering allowing the user to freely extend the LMS with additional functions. For the first technical issue, we propose a construction model for new functions based on clauses and equivalent transform rules. In a word, this is a construction methodology for Web applications that inclusively describes the database technologies, network connection, and WWW technologies. In comparison to other construction methods, the user can specify the system by a lower cost using this methodology.

The second technical (security) issue is solved by permitting only users who are listed on a specified menu to execute the program. We solve these technical issues by instituting a mechanism which allows safely, automatic database access.

Additionally, since a constructed function can be shared and published by other users, users can use copies of functions constructed by any other user. A user can construct new function based on that copy, so the LMS can evolve by extending it with many such functions. In developing this evolutionary e-learning system, it is important that it includes a simple construction model independent of any particular Web technology, database technology or network environment. In addition it should have a safety mechanism for connecting to a database, and a framework for sharing and expanding functions. In section 2, the difficulties and security risks involved when permitting the user to freely make functions for the system are discussed. Additionally, the section proposes a simple construction model for resolving these problems, and discusses the safety of the model.

In section 3 we outline the mechanism for implementing a program specified by the proposed construction model introduced in Section 2, and a framework for executing the model on the system. Additionally, the section discusses the effectiveness of the model, and a structure for sharing and publishing constructed programs.

II. Clause-Rule Model

A. Challenges Faced by the User in Creating Functions

An e-learning system is a Web application. The basic mechanism of a Web application is data communication between client and server via a network connection. The basic components are a web browser, a CGI[2] script, a database and a network. The main components of an e-Learning system are a CGI script and the database access functions via a network connection. Fig. 1 shows the relationship of these components. A Web browser has two main functions: 1. to display HTML documents obtained from a server, and 2. to send a request for data to a server. The CGI script operates in accordance with the request from the client. It accesses the database and dynamically composes documents from the data and returns them to the client.

K.Nakamura is with the Faculty of Computer Science, Hokkaido University, Sapporo, Hokkaido, 060-0811 Japan e-mail:knakamura@ist.hokudai.ac.jp.
K.Akama is with the Information Initiative Center, Hokkaido University, Sapporo, Hokkaido, 060-0811 Japan e-mail:akama@iic.hokudai.ac.jp.
H.Mabuchi is with the Faculty of Software and Information Science, Iwate Prefectural University, Takizawa, Iwate, 020-0193 Japan e-mail:mabu@soft.iwate-pu.ac.jp.

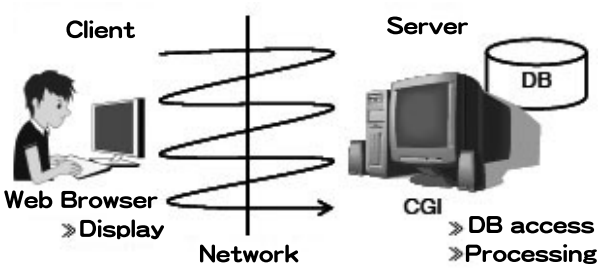


Fig. 1. the mechanism of a Web Application

This mechanism is the basic operation of a Web application. However, it is also the reason why knowledge of network technology and database operation are essential in order to construct a properly operating system.

As a result knowledge of HTML for interface specification, database query model for database management, and Web and network related knowledge for Web application construction, are indispensable.

As depicted in fig.2 the developer uses a script which to specify these varied models.

It is very difficult to simplify the function creation process for the user because it requires knowledge of several Web technologies.

In addition, a developer must consider security in the database and there is an anxiety in make construct the new function by the viewpoint of the safety management of the system. Moreover, from the viewpoint of system management, there is a need to consider security of the database, even while allowing the user to make the function. If a simple construction framework for LMS functions was provided for students and teachers, then everyone would be able to construct new LMS functions based on individual needs.

We propose the Clause-Rule model for such a framework.

B. Modeling of Web application by clause set and rule

Using this model knowledge concerning the Web application technologies is not necessary and the security of the database is guaranteed, when the user constructs the LMS functions. Because an e-Learning system consists of various models, such as interface description model, database query model, etc., the difficulty , for the user, of creating functions increases. Then, it thinks about the method of simply describing a new function of the LMS. Therefore we need to come up with a simple means of specifying the system. An e-Learning system's function can be specified using clause sets and rules. For example, the database can be specified using clause sets. Also, rules can be used to process the clause sets. These clause sets and rules specify the essence of the function.

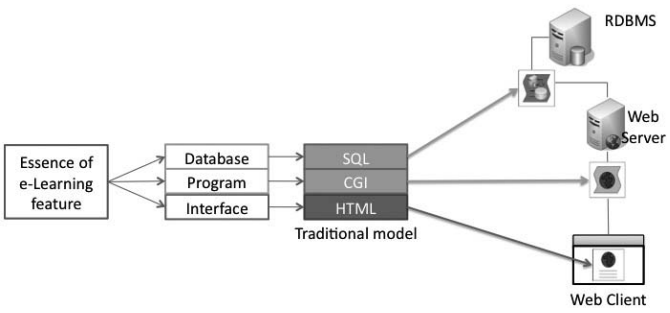


Fig. 2. Traditional Model

C. To describe the DB access via definite clause

Definite clauses can be used to model the database. The table of a database corresponds to an atom and database query corresponds to logical computation.

As shown in fig.3, by converting definite clauses via N-rules and using logical computation on the resulting clauses, access to the database is possible.

Also, the N-rule can process definite clauses. The table of the database corresponds to an atom and database invocation corresponds by the logical calculation. Thus, the N rule can operate the database by the definite clause. The purpose of the reason to use N rule is to obtain two or more solutions in the data base access that uses a definite-clause. To access the data base of N rule is describe as follows: D atom named getScore is executed as shown in line cl_1. The rule that describes D atom is shown since line cl_2. The getScore atom is rewritten in the rule that executes the lms:db atom, and acquired from the Score table in S-expression. The database table can be modeled by expressing the list according to the number of tupples of tables by S-expression.

cl_1 : $H_s \Rightarrow \{(getScore\ ^*list)\}, B_s.$
cl_2 : (getScore$^*list) \rightarrow$
cl_3 : (lms:db(table:select Score $(^*id\ ^*qid\ ^*score))^*list)$.

This DB access method is not as efficient as SQL in data processing, but it gives a clear, simple description in S-expression. The specification-program can describe the essence quite simply.

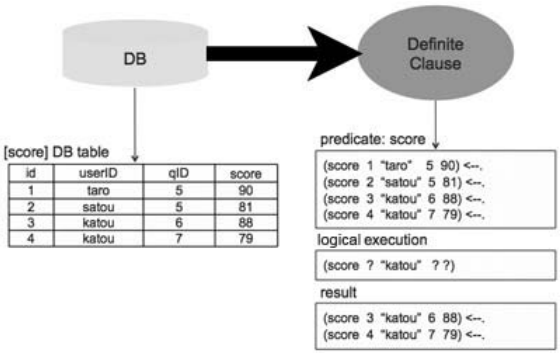


Fig. 3. Logical DB access

By applying data selection rules to the clause sets data can be extracted. For example, to produce student individual data, the information can be derived from the clause sets. When the database is represented using clause sets, N-rules can be used to extract the information it contains. An N-rule is one type of equivalent transform rules based on ET theory[5][6].

Combining N-rules with D-rules enables us to achieve flexible data processing and realize bot sides of Web application functioning. An interpreter for application of the S-expression and rules in the e-Learning system already exists. As shown in fig.4, the specified essence is read by the interpreter and automatically converted into separate functions. This is called the Clause-Rule model.

D. Syntax of the Clause-Rule Model

1) N-rule Syntax: The Clause-rule model can specify using N-rule syntax. An N-rule is composed of a head part(H_s), a condition part(C), an execution part(E_n) and a body part(B_{s_n}). It has the following format;

$$\begin{aligned} H_s,\{\mathrm{C}\} &\Longrightarrow \{E_1\},\ B_{s_1};\\ &\Longrightarrow \{E_2\},\ B_{s_2};\\ &\ \ \vdots\\ &\Longrightarrow \{E_n\},\ B_{s_n}. \end{aligned}$$

A head part consists of one or more atoms while all other parts consist of zero or more atoms. When a head is transformed into two or more bodies as shown above, ";" is used for dividing those bodies. If the head is transformed to only one body, such an N-rule is described as shown in the following format;

$$H_s,\{\mathrm{C}\} \Longrightarrow \{\mathrm{E}\},\ B_s.$$

Each part is composed of atoms and executes as follows;

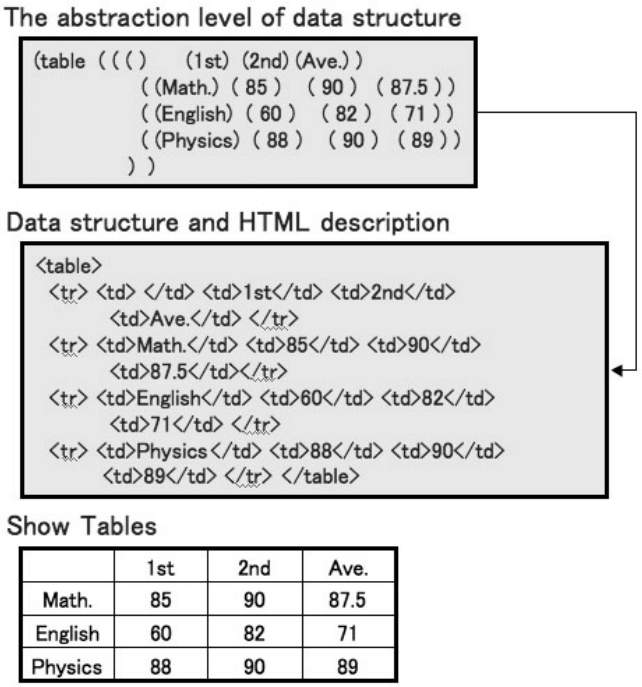
1) Head atom matching
2) Cond atom execution
3) Exec atom execution
4) Conversion to replacement atom

The condition and execution parts consist of zero or more atoms. In the case where there are zero atoms these parts are deleted. Variables in a rule begin with the asterisk '*' symbol.

N-rule syntax can describe flow of computation, user interface definition, and data display, in a natural, intuitive way. As depicted in cl_1, the program starts from (S init). The getdb section specifies database access. This is detailed in II-D3. Next the data acquired from the database is processed as a list specified in S-expression. The list in S-expression simply defines a table as is, and can be outputted to the user interface. The details are shown in **??** Next, data is passed to the user interface (display device) as shown in $cl_2$and it is displayed on the screen. In addition, the button to accept the next request is shown to the user.

$$\begin{aligned} cl_1 :&\ \ (S\ init) \Rightarrow \{getdb(d), makelist(d, d_1)\}, (C\ send(d_1)).\\ cl_2 :&\ \ (C\ recv(d_1)), \{disp(d_1), setClick(c)\} \Rightarrow (S\ send(c)).\\ cl_3 :&\ \ getdb(d) \rightarrow \ldots (reffer\ \text{II-D3})\\ cl_4 :&\ \ makelist(d, d_1) \rightarrow \ldots (reffer \text{II-D2}) \end{aligned}$$

2) Specifying Tables using S-Expression Lists: For the Clause-rule model, the page can be presented in table form by composing the data. As depicted in Fig.5 the table can be expressed by a simple S-expression type list. The S-expression type list can define the logical structure, etc., of the data necessary to express the table.



	1st	2nd	Ave.
Math.	85	90	87.5
English	60	82	71
Physics	88	90	89

Fig. 5. Description Table by S-exp.list

A Web browser displays data tables constructed in HTML. Therefore, we need to come up with a method of transforming the data in the S-expression list to HTML. For example, the description of a table is shown in Fig.5. The S-expression list can express the table structure in the same way HTML tags do. The S-expression list can describe the output page without us having to think about the HTML structure because the list structure corresponds to the HTML format.

The advantage of using list processing lies in the fact that columns and rows can be flexibly added. For example, the description of a simple specification program is shown in Table I. The transformation of this program is shown in fig.6.

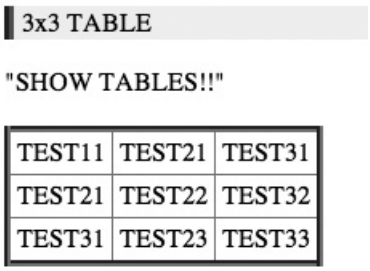


Fig. 6. Transformation of 3x3 Table

3) Specifying Database Access using Clause Sets and N-Rules: A database can be modelled using clause sets. By using N-rules to treat clause sets, a logical method of processing is proffered. When compared to SQL, the efficiency of this

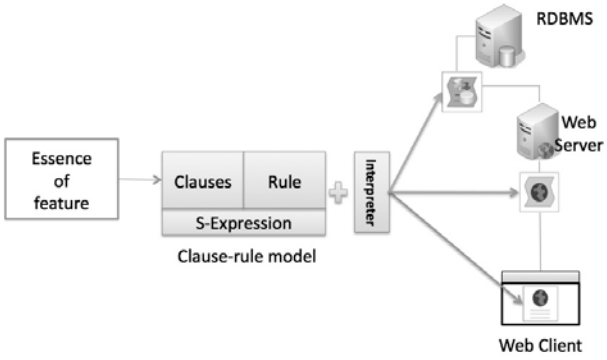


Fig. 4. Clause-rule model

TABLE I
PROGRAM LIST OF 3 X 3 TABLE

line 1	: '(S init)' atom is starting.
line 2	: A data make in the server and send to the client.
line 6	: The Textdata make on the server.
line 7-10	: The Tabledata (list) make on the server.
line 13	: The Data show at the client.

```
1:    (S init)
2:    ⟹ { (table2 *data)}, (C showT *data).
3:
4:    (table2 *data)
5:    ⟶(lms:setData *data(
6:          (text "SHOW TABLE!!")
7:          (table (
8:                ((text TEST11) (text TEST21) (text TEST31))
9:                ((text TEST21) (text TEST22) (text TEST32))
10:               ((text TEST31) (text TEST23) (text TEST33))))
11:         )).
12:
13:   (C showT *data),{(lms:print *data)}⟹.
```

method is low however the framework offers a simple, intuitive way of treating data. By processing the data structure using N-rules, actual RDB tables need not be considered and database access can be flexibly specified.

cl_1 : $(S\ init) \Rightarrow \{getdb(d), makelist(d, d_1)\}, (C\ send(d_1))$.
cl_2 : $(C\ recv(d_1)), \{disp(d_1), setClick(c)\} \Rightarrow (S\ send(c))$.
cl_3 : $getdb(d) \rightarrow \ldots (reffer$ II-D3)
cl_4 : $makelist(d, d_1) \rightarrow \ldots (reffer$II-D2)

4) D rule: The data acquired from the data base is processed, and the following is the output as a table corresponding to S-expression list. The condition and execution parts are described using D rules[5]. The syntax of D rule is as shown below:

(Head atom), {(Cond [zero or several atoms....])}
⟶ (Body atom).

D-rules specify flexible database access and processing of the data returned to the Web browser.

III. LMS FUNCTION CONSTRUCTION SYSTEM FOR EXTENDING

A. Features

In the following sections we will explain the features of just such a new function framework that was developed by us.

The features are:

- (The system) interprets specifications via Interpreter CGI
- facilitates sharing of specifications
- finds convenient functions

B. Menu

The function of the menu shown in fig.7 is to make and to edit the new features made and included (program), and manage shares and user authentication. The menu facilitates the management of the program by acting as an intermediary between the new features made by the user and the Clause-rule model interpreter. In addition, the process of user authentication for a safe database access is automated.

1) User's Authentication: The menu accesses the table in the database to manage user information, and guarantees that a certified user is using the program. The menu can understand who pushed the execution button of the program by internal processing as shown in fig.8.

2) Program Launcher(Button): The program launcher is a button to start the program with the e-Learning system. There is a button that corresponds from Menu1 to Menu3 as shown in fig.9. The new features included managed with the repository is allocated in the button, and it is possible to use it with LMS.

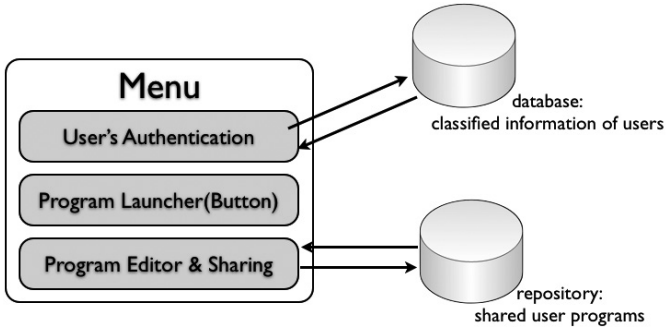


Fig. 7. Menu:feature manager

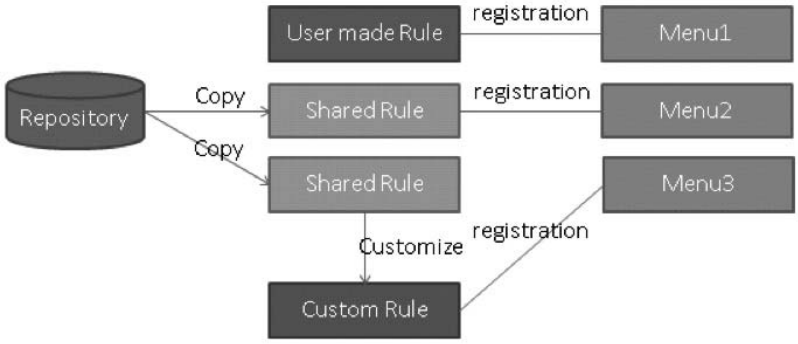


Fig. 9. Program shared system

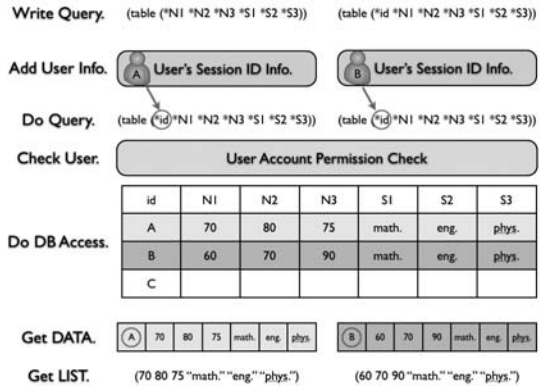


Fig. 8. User Authentication

3) Program Editor and Sharing: The program editor can edit the program. In addition, the program that I made can be shared in the repository. Certainly, programs made by other users can also be used.

C. Interpreter

We will now explain the mechanism by which the menu allows the program to be executed. The proposed model is executed as shown in Fig.10 by the interpreter acting as a Web application. When the menu button is pushed, the corresponding program in the program launcher is executed.

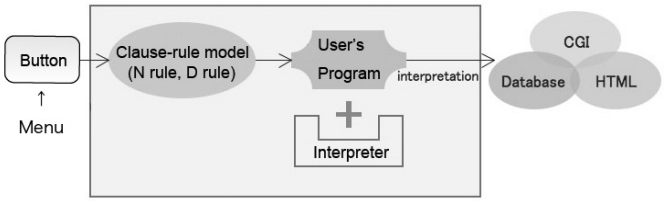


Fig. 10. Interpreter

1) The Interpreter CGI for Specification execution: In this section, we explain the execution environment of interpreter CGI. The execution environment loads the interpreted specification program as input data and executes it like a CGI program. The user selects the specification program from the database and can execute the selected program. We now explain the execution process shown in fig.11:

1) User starts access to server. (by Client)
2) Acquire the specification program from a database. (by Server) Specification is interpreted by Interpreter CGI. (by Server)
3) Interpreted data generates HTML of execution result at execution environment.
(by Server) Output to Web browser (by Client)

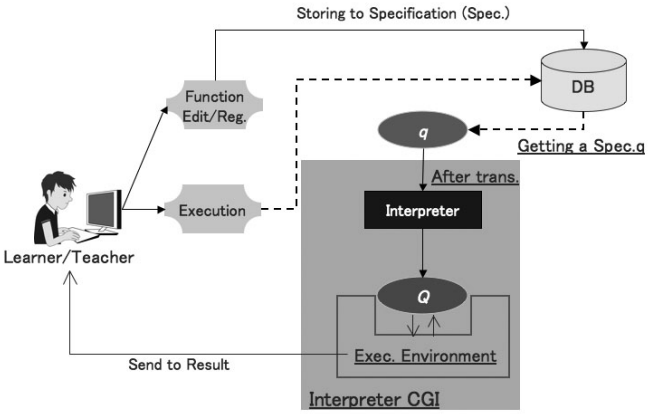


Fig. 11. Execute Specification

D. The Repository and Program Sharing

All programs are made from S-expression and the rule in this system as shown in Fig.9, and it is accumulated and shared in the repository. The program put on the repository can privately use the one that I made. Moreover, other users can use it because it shares. The program acquired from the repository can be customized. In addition, the customized program can itself be shared.

IV. Discussion

A. Clause-rule Model

It was necessary to mount the Web application in a past model based on the knowledge of an individual elemental

technology, and it was necessary to assume Web, the data base system, and the network. On the other hand, they are inclusively modeled by S-expression in the Clause-rule model, and it is possible to execute it as a program by the clause and the rule. The Clause-rule model is Web application construction model where the essence of the function can be inclusively described by the clause and the rule as for an individual function like string data, the data base access, and the network, etc. that are the elemental technologies in the Web application. The Clause-rule model goes flexibly in a logical access method to the data base compared with the model of a procedural model and an object-oriented model, and facilitates the treatment of flexible string data in the list processing. Moreover, a logical access method can be flexibly execute as well as the model of a procedural model and an object-oriented model compared with the construction model of the list processing like a functional programming language. It is difficult to describe going side by side requested from the Web application though the definite clause model of the data base can be similarly described when thinking about modeling the Web application in Prolog. In addition, the Clause-rule model can be made while piling the rule though Prolog should describe the entire Web application at a time by a definite clause and the rule. This is convenient in making of each Web function a component. The Clause-rule model has the feature of catching the whole image of the Web application specification easily by describing the essence of an inclusive function by the N rule, and calling the component of each Web function in the ET D rule for such reasons.

B. Concurrency and Clause-rule model

In the Web application, correspondence to the data base access and the rich Internet application is indispensable. There is going side by side in the data base access and the Rich Internet application, and the frame that can describe such going side by side is important. The Clause-rule model enables correspondence to such going side by side by using N rule where going side by side can be described the clause by the rule. For the Clause-rule model, going side by side can be expressed as follows by a final paragraph by showing of one atom one client:

$$cl_1 : \quad (Websystem) \leftarrow (ClientA), (ClientB), \ldots .$$

C. Security

It now becomes possible to automate the user authentication process by utilizing the mechanism provided by the menu to call the program, and to execute the program safely. The user doesn't have knowledge concerning the security of the system. Therefore, the method to access the database safely via N rule was offered. Using this method, it is now easy to make functions as long as the Clause-rule model and N rule are understood.

D. Function Sharing

The sharing function is a mechanism to share the learning supporting function with very high utility in LMS. The teacher who doesn't have the knowledge of the programming can use the share function. Moreover, if it knows the Clause-rule model and syntax to some degree, the study support function can be constructed for myself, and it be shared. Moreover, the shared learning supporting function can be voluntarily improved. The mechanism of such sharing and the improvement can evolve the function in LMS. This developer makes a big framework of LMS, and the user can also make a detailed function of LMS. In the purpose of making the user make the function of LMS simply and safely, the Clause-rule model is an extremely important method.

E. A component-based Web programming

A D rule is built-in and a component of the basic Web function. Now it discuss a component of Web function. Since software systems become larger and more complex, the need for the development of cost-effective and high quality software in a short period of time increases accordingly. As a result, there has been increasing attention to component-wise program generation[3][4][7]. In the development of the web application, there is a component-based method shown in Figure 12. To achieve this method, the object-oriented language is adopted well. Since a lot of reuse knowhows are shared and it is used well. The purpose of this is with the feature that the behavior of the calculation execution by the component division and the message passing is managed easily and it is easy to reuse. The combination between components exists hugely. As for the check on all models, and the guarantee of the correctness when components combine, these are high cost. It introduces the associated study on the specification-program in the component-based program construction. The ET rule is excellent as the component technologies. Since ET rules have complete independence with regard to partial correctness, adding new ET rules to an existing ET rule set does not impair the correctness of the existing ET rule set. Therefore, the ET rule can create a large scale and correct program by adding a small program (new ET rule) on existing ET rule set. In response, the squeeze method[**?**] where a program is created by successively accumulating ET rules has been proposed. Since ET rules can be considered as components constituting a program, also when viewed from the perspective of component-wise program generation, they can confer considerable advantage[8]. I want to discuss the applicability to the Web component that the specification-program uses as a library as future tasks.

F. Future Tasks: Compiler CGI

Presently, interpreter CGI can interpret the specification. However,high speed calculation is difficult. The Compiler CGI can transform a fast CGI beforehand from the specification program. At that time, it can construct the e-Learning system from the specification program.

V. Conclusion

In this paper, we suggested an evolvable e-Learning system via a specification program and constructed an LMS function

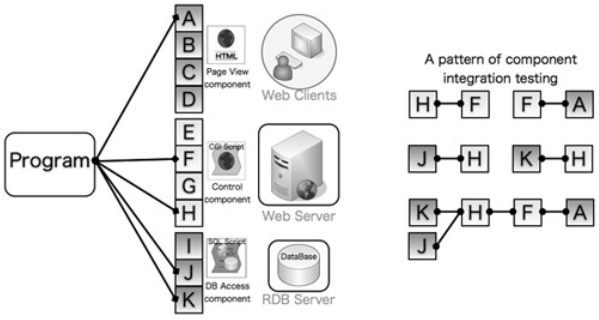


Fig. 12. A component-based Web programming

construction system. Also, we proposed the necessary requirements for new functions. We showed the challenges facing free function construction proposed the Clause-rule model as the solution. Thereupon, we explained the structure of Interpreter CGI and sharing of the specification program. In addition, we extended the execution environment of the specification program for the e-Learning system. If a user can construct this function, the system's functionality increases based on individual needs. If the function can be shared by a user, the whole system evolves.

REFERENCES

[1] Dougiamas, M. and Taylor, P.C., Moodle: Using Learning Communities to Create an Open Source Course Management System. Proceedings of the EDMEDIA 2003 Conference,pp.171-178, Honolulu, Hawaii, 2003.
[2] D. Robinson, K. Coar, The Common Gateway Interface (CGI) Version 1.1, Network Working Group, RFC3875, Category: Informational, 2004.
[3] Hopkins,J., Component primer, Communications of the ACM, vol.43, no.10, pp.27-30, 2000.
[4] IBM San Francisce, Concepts and facilities, IBM Corporation, 1977 Project, Software Development, vol.6, no.2, 1998.
[5] K.Akama, H.Koike and H.Mabuchi, A Theoretical Foundation of Program Synthesis by Equivalent Transformation, Perspectives of System Informatics, Lecture Notes in Computer Science ,Vol.2244, pp.131-139, Springer Verlag, Heidelberg, 2001.
[6] Koike,H., K.Akama and E.Boyd, Program synthesis by generating equivalent transformation rules, Proc. of the 2nd International Conference on Intelligent Technologies, Bangkok, Thailand, pp.250-259, 2001.
[7] Szyperski,C., Component software: Beyond object-oriented programming, Addison-Wesley, 1999.
[8] Mabuchi,H., K.Akama and T.Wakatsuki, Equivalent transformation rules as components of programs, International Journal of Innovative Computing, Information & Control, vol.3, no.3, pp.685-696, 2007.

Keisuke Nakamura Affilication:
Fuculty of Computer Science, Hokkaido University

Address:
North 11, West 5, Kita-ku, Sapporo, Hokkaido, 060-0811 Japan

Brief Biographical History:
2001 - Faculty of Bussiness Admin. and Information Science, Hokkaido Information University
2005 - Graduate School of Hokkaido Information University
2007 - Faculty of Computer Science, Hokkaido University

Membership in Academic Societies:
Information Processing Society of Japan(IPSJ)

Kiyoshi Akama Affiliation:
Professor, Division of Large-Scale Computational Systems, Information Initiative Center, Hokkaido University

Address:
North 11, West 5, Kita-ku, Sapporo, Hokkaido, Japan
Brief Biographical History:
1984.4 - Associate Professor, Faculty of Engineering, Hokkaido University
1999.4 - Professor, Center for Information and Multimedia Studies, Hokkaido University
2003.4-Professor, Information Initiative Center, Hokkaido University
Main Works:
"Formalization of Computation Models in View of Program Synthesis," Proc. of the 4th International Conference on Intelligent Technologies (InTech 2003), Chiang Mai, Thailand, pp.507-516, 2003.
Membership in Academic Societies:
The Japanese Society for Artificial Intelligence (JSAI)
Information Processing Society of Japan (IPSJ)

Hiroshi Mabuchi Affiliation:
Fuculty of Software and Information Science, Iwate Prefectural University

Address:
152-52, Sugo, Takizawa, Iwate 020-0193, Japan
Brief Biographical History:
1995 - Lecturer, Tohwa University
1998 - Lecturer, Iwate Prefectural University
2002 - Associate Professer, Iwate Prefectural University
Main Works:
"Equivalent Transformation of Member Constraints on Interval-Variable Domain,"
Journal of the Japanese Society for Artificial Intelligence, Vol.17, No.1 C, pp.23-31, 2002.
Membership in Academic Societies:
The Japanese Society for Artificial Intelligence (JSAI)