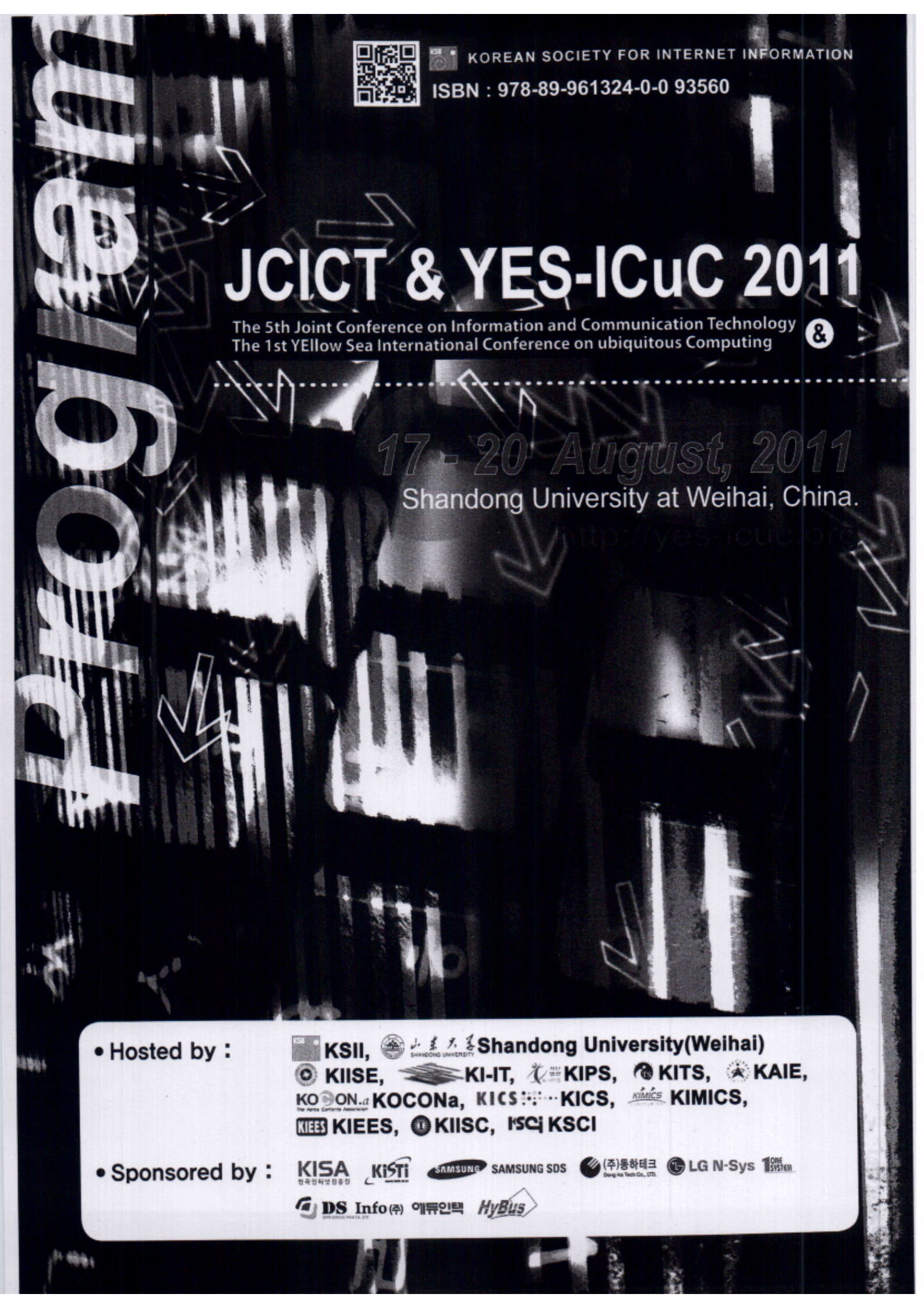

KOREAN SOCIETY FOR INTERNET INFORMATION
ISBN : 978-89-961324-0-0 93560

JCICT & YES-ICuC 2011

The 5th Joint Conference on Information and Communication Technology &
The 1st YEllow Sea International Conference on ubiquitous Computing

17 - 20 August, 2011

Shandong University at Weihai, China.

http://yes-icuc.org

• Hosted by : KSII, Shandong University(Weihai), KIISE, KI-IT, KIPS, KITS, KAIE, KOCONa, KICS, KIMICS, KIEES, KIISC, KSCI

• Sponsored by : KISA, KISTI, SAMSUNG SDS, (주)동하테크, LG N-Sys, DS Info(주), 에듀인텍, HyBus

Title & Abstract Pages

JCICT &The first Yellow Sea International Conference on Ubiquitous Computing (YES-ICUC) 2011, August 2011

A Study on Data Architecture of Core Design Code for Efficient Loading Pattern

So Young Moon[1,2], Sung Tae Yang[3], Young Suk Jung[2], R.Young Chul Kim[1]
[1] Software Engineering, Hongik University
Seoul, 339-800 - KOREA
[e-mail: whit2@selab.hongik.ac.kr, bob@hongik.ac.kr]
[2] Nuclear Information Technology, ACT (Atomic Creative Technology)
Daejeon, 305-509 - KOREA
[e-mail: whit2@actbest.com, youngsuky@actbest.com]
[3] Operation Technology, KHNP Nuclear Engineering and Technology Institute
Daejeon, 305-343 - KOREA
[e-mail: fuse@khnp.co.kr]
*Corresponding author: So Young Moon

Abstract

Selection of loading patterns affects cost and safety in reload design at nuclear power plants. Core design data inputs and outputs of text-format UNIX code are processed manually. Selection of labor intensive loading patterns can negatively influence efficiency. This paper proposes data architecture for an efficient loading pattern selection to resolve these issues. It visually depicts tables, graphs, and 2D images by generating input automatically and processing text-typed outputs through GUI-based web pages. Repetitive tasks are minimized for users, so they can focus on engineering decision-making for optimal loading pattern selection, thereby increasing effectiveness.

Keywords: Loading Pattern Search, Architecture, Data Processing

1. Introduction

Scientific results obtained from UNIX-platform applications are generally text-based. This process is labor intensive, requiring extensive and time-consuming raw data analysis. Inputs and outputs for nuclear power plant pressurized water reactor core design code run on UNIX tend to be text-based, and run by console. Software for nuclear power plant core design, which produces design and operation data, can contribute to the safe, effective and efficient operation of a nuclear power plant over 18 months intervals [1]. In selecting a loading pattern, the applicable fuel information and the current cycle loading pattern must be taken into account. The resulting selection of a loading pattern must satisfy the primary design limit parameters and the core design requirements, and must consider cost and safety issues. For text-based results analysis, the primary core parameters from the code results must be assembled, and data translation must be performed manually. As a result, data analysis takes about 40 percent of total design time. For the selection of efficient loading patterns, we suggest a core design data architecture that seeks to improve these conditions. Repetitive tasks are minimized for users, allowing them to focus on engineering decision-making for optimal loading pattern selection, thereby increasing effectiveness [2].

This paper consists of five parts. In Part 2, input and output data and a CGI-based architecture for running the core design code are examined. In

Part 3, a data architecture for core design that will allow the selection of efficient loading patterns is suggested. In Part 4, an analysis of the core design code, which employs RIA technology-based web pages is presented. Finally, in Part 5, conclusions and possibilities for further research are proposed.

2. Related Work

2.1 Data of Core Design Code

The core design codes of pressurized water reactors consist of either the Westinghouse type of the OPR1000 type. The ANC is for Westinghouse types. A shuffling code program describes the reactor, performs reactor computation, and models the grid. It describes various reactor facilities such as the arrangement of the fuel assembly, control rods, incore instrumentation, and fuel rod design. It also includes the computation, and results of code production. The results of code production include various parameters about the core. The results analysis is called the loading pattern selection. As can be seen in Table 1, about forty thousand lines of input data and output data are analyzed manually for selection of an optimized loading pattern that satisfies the core design requirements.

Table 1. Format of shuffling code Input/Output

Input

Shuffling

```
……
...omit...
……
* SHUFFLING
720807,                    12, 1,2,44, 2,0, 8, 3,2
+  ,              13, 4,2,              3, 8,0
+  ,          11, 9,2,      5,11,1, 6,12,2, 7,13,0
+  ,          39,15,2,                  10,18,2,
27,20,0,23,21,1
+  ,              22,23,3,16,24,2,17,25,0, 2,26,3,
34,28,2
+  ,31,29,2,     30,31,2,     37,33,1,     19,35,1
+        ,24,37,1,                 38,39,0,41,40,0,
32,42,3,36,43,2,14,44,0
+      , 8,45,1, 3,46,3,           23,48,0,34,49,1,
14,51,3,18,52,0
……
...omit...
……
```

code

```
……
...omit...
……
900060,HRS=650.134,PPM=1174.0, EFF=1.00472
900070,HRS=650.134,PPM=1103.0, EFF=1.00473
900080,HRS=650.134,PPM=1035.0, EFF=1.00474
900090,HRS=650.134,PPM=971.0, KEFF=1.00475
900100,HRS=650.134,PPM=911.0, KEFF=1.00476
900110,HRS=650.134,PPM=846.0, KEFF=1.00477
900120,HRS=650.134,PPM=772.0, KEFF=1.00478
……
...omit...
……
```

Output

```
……
...omit...
……

ASSEMBLY BATCH                    1 J1  2 J6  3 J4
REL.POWER DENS/CORE               .2745 .3173 .3259
AVERAGE BURNUP(MWD/T)             39328 39900 39763
AVERAGE K-INFINITY                .9066 .8899 .8898

                  9 J0  10 L1  11 K0  12 K6  13 K6  14 L6
                 .4315 1.0767 1.2393 1.2103 1.1569 1.2915
                 36439     0 16153   18609   20173     0
                 .9352 1.1298 1.0661 1.0302 1.0200 1.0792
...omit...
……
```

2.2 CGI based Architecture for Executing UNIX program on Web Environment

If users with None UNIX system experience want to work in this environment, they should study the usage of the system beforehand. We have already suggested a CGI-based architecture for executing the UNIX program in the web environment that will solve any inconveniences. First of all, an Apache server should be installed in a UNIX system for the CGI configuration. This server is programmed in C language. As soon as the CGI programs finish code running the code, the text-based outputs analysis begins. The CGI-based architecture system inputs results by invoking the functions from WAS (Web Application Server) [2].

3. Data Architecture of Core Design Code

3.1 Data Architecture

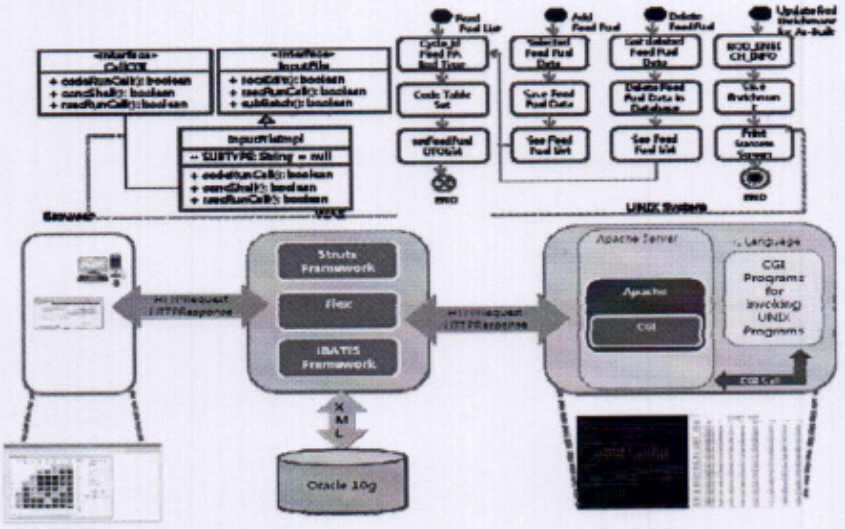

Fig. 1 Architecture for Data of Core Design Code

Fig. 1 shows the data architecture for the core design code data, which is extended by a CGI-based architecture for executing programs in UNIX. It visually depicts charts, data-grids, and trees with RIA(Rich Internet Application) for result data checking. It performs parsing of the text-based results, and saves the data to database. It uses an iBatis framework (SQL mapping framework) for querying and adopts Flex for the visualizing of data. A parsing class is created that saves raw data to the database. It gives correct information rapidly by use of a framework such as Struts, Flex, and iBatis. A user clicks a code-run button in the web-browser, and then the WAS performs the score design code by invoking functions in UNIX through HTTP.

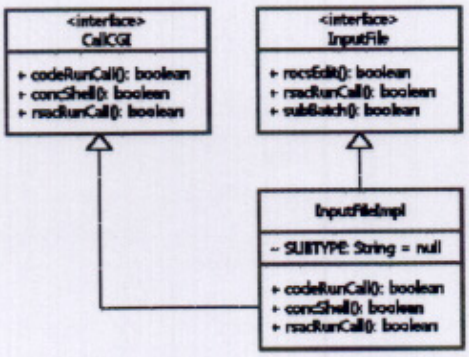

Fig. 2 Class Diagram for the code-run

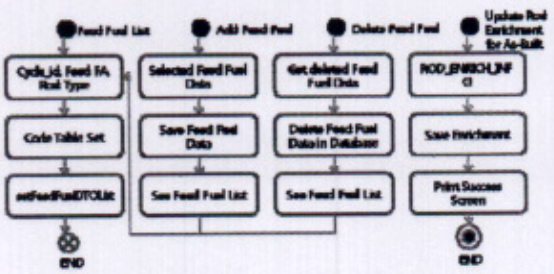

Fig. 3 Activity diagram for feed fuel process

3.2 Data Architecture Implementation

The suggested data architecture presents basic scenarios with UML (Unified Modeling Language) notation such as class diagrams, activity diagrams, and sequence diagrams.

Fig. 2 shows a class diagram for the code-run module. It calls codeRunCall () in InputFileImpl and then performs the core design code, finally responses results are sent to users with messages.

Fig. 3 shows an activity diagram of a feed fuel process component of data architecture for workflow.

Table 2. iBATIS statement for number of plane

```
..... Omit .....
<![CDATA[
SELECT DISTINCT c.plane_no
FROM   core_layout a, plant b,
         result_core_man c,
       job d, result e,
       core_lp f, core_type g
WHERE d.job_id = c.job_id
AND   d.job_id = e.job_id
AND   b.plant_id    = f.plant_id
AND   e.maneuver_no = c.maneuver_no
..... Omit .....
```

Table 2 An iBatis sentence for checking a code result for a fuel rod

Table 3. Flex Statement for Chart

```
<mx:horizontalAxis
<mx:LinearAxis id="Bu2Axis"
title="Cycle Burnup(HWD/MTU)"
minimum="{BUMIN}"
displayName="Burnup" />
</mx:horizontalAxis>
displayName="Fq"/></mx:verticalAxis>
```

Table 3 shows flex code to visualize data with chart from iBatis and struts.

4. Case Study

The suggested data architecture of the core design code for efficient loading pattern selection is used to process the results of the core design code. If it is applied to a system, users will not need to analyze text-based results. They can analyze results of the entire core design with the flex-based web pages.

We test the core parameters of a 1/4 core through the data architecture for a loading pattern selection with a 1/4 core image, Results are shown in Fig. 4. This method offers more convenience and is faster.

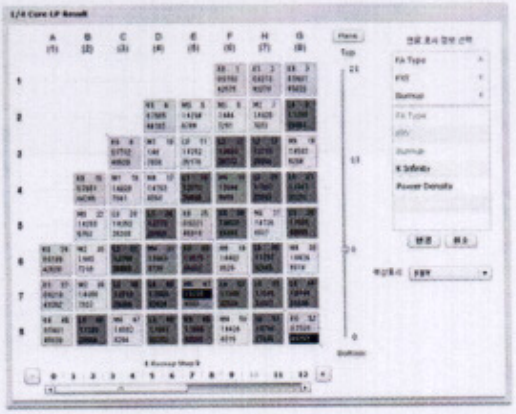

Fig. 4 Result of 1/4 Core Loading Pattern

In **Fig. 5**, a Flex-based web-page shows charts and values for primary core parameters per burn-up.

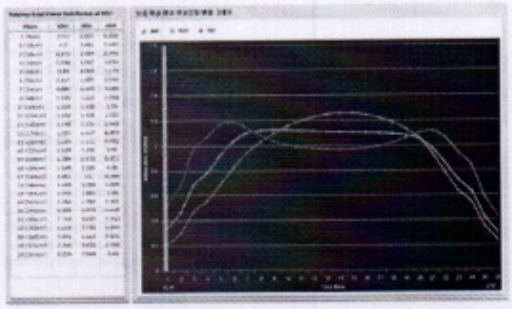

Fig. 5 Chart of Core Parameters per Burnup

5. Conclusions

For the next cycle, the code needs to arrange efficient fuel insertion through the spend style of a core loading pattern of the current cycle. In this paper, we have suggested a data architecture that processes text-based results in order to select an efficient loading pattern. It processes results of the core design code automatically, and then saves the whole data set to the database. Therefore it visually depicts tables, graphs, and 2D images by generating input automatically and processing text-typed outputs through GUI-based web pages.

This time we present data for only 1/4 core, we will determine data for a full core with 3D graphics in the future.

References

[1] Sung Tae Yang, Hyeong Jin Kim, "Unified Core Management System Development for PWR," in *KPIC*, 2009.

[2] So Young Moon, Young Suk Jung, Hyeong Jin Kim, Chae Yun Seo and R.Young Chul Kim, "A Study on CGI Based Architecture For Executing Core Design Code on Web Environment," *KIPS*, vo.17, no.2, pp.345-349, 2010.