E-Learning Distributed Cloud Built on MVC Design Patterns for Service Task Management

Ruth Cortez
University of Aizu
Aizu-wakamatsu, Fukushima
965-8580, Japan
m5132203@u-aizu.ac.jp

Sidhant Rajam
University of Aizu
Aizu-wakamatsu, Fukushima
965-8580, Japan
m5132202@u-aizu.ac.jp

Subhash Bhalla
University of Aizu
Aizu-wakamatsu, Fukushima
965-8580, Japan
bhalla@u-aizu.ac.jp

Alexander Vazhenin
University of Aizu
Aizu-wakamatsu, Fukushima
965-8580, Japan
vazhenin@u-aizu.ac.jp

ABSTRACT
Cloud computing is prevalently influencing the present day business trends and operations. Due to Cloud's enticing features, that can be measured by the ease in which the services are delivered to a range of consumer demands without changing the underlying technological implementations or the economics associated with the systems. The e-Learning arena is also not an exception to distance itself away from Cloud driven computing technology. The key point in the e-Learning environment is to process any type of diverse problem within a moderate response time across ubiquitous network channel and securely deliver the output to the authorized user on the most user-friendly interface. The bottom-line of our research is to provide a Software Development Platform enabling the clients to specify and design their task that can be hosted as a service through the Cloud which is interfaced by Web services technology. We are proposing an e-Learning enterprise Cloud based on the Model-View-Controller design patterns to expose the reusable View and Controller tiers as Distributed Cloud, each one wrapped individually in Web services. The Controller Cloud implements any task as a configurable entity, which is individually exposed as a service, and/or simultaneously uses third party Cloud services to execute the most complex problems. The loose coupling between View and Controller potentially benefits to baseline and manipulate the desired level of system's Quality of Service in terms of performance, scalability and security.

General Terms
Design, Human Factors and Verification.

Keywords

1. INTRODUCTION
Traditional classroom methodology usually consists of a set of activities that teacher should perform to transfer knowledge to students and develop the skills to apply this knowledge. Under traditional scheme teachers find difficult to focus on individual needs. Students usually play a passive role and teacher's effort is mostly concentrated on course organization, preparation, delivery and scoring assessments. Eventually, this might lead into resources challenges when education demand increases in a higher pace than the resources of education providers (e.g. professors, teachers, facilities, etc). The use of Web technologies such as multimedia, VoIP, chat, blogs, wikis, computational processing and the convenience of having the information available “anytime, anywhere”, are some of the potential advantages of incorporating e-Learning systems in education, and by this way overcoming traditional classroom challenges [7]. Accordingly, the activities are transitioned from being teacher centric to student centric and increases student motivation by playing a more active role during education process.

It is possible to distinguish several approaches to implement an e-Learning environment. The main idea of the first approach was essentially transition contents of traditional class to a Web application and made available the course information to students, but paying low attention to the pedagogical work flow [10]. The second approach is called as a Blended Learning, which incorporates the usage of e-Learning applications to traditional classroom activities, using various delivery methods such as face-to-face lecture, online material for self study, video, audio, tests, surveys, simulations, workshops, Web seminars, discussion forum etc. The advantage of Blended Learning is more flexibility adapting to student’s needs due to the combination of Synchronous physical formats, Synchronous online formats and Self-paced Asynchronous formats, a detail description can be

Categories and Subject Descriptors
K.3.1 [Computers and Education]: Computer Uses in Education – Collaborative learning, Distance learning.
found in [10]. Finally, it introduces tools for student’s performance analysis and support. The Flex-eL e-Learning environment [7] incorporates most of the advantages of blended learning plus the focus on learning process work flow. In this environment, the course content is divided into modules and represented graphically for better comprehension. It includes the usage of collaboration tools making possible, for example, the ability to identify students that are working on the same activity and promoting collaboration groups. In this specific example, the environment is designed to allow students manage their individual time, more tailored learning, flexible pathways in which the time constraints depends on student time, all of this towards the idea of its usage as Virtual University.

The last approach discussed in our review considers the e-Learning environment as a set of Web services. It incorporates the main benefits described on previous approaches but envisioning the environment with the following requirements: open architecture and interface, durability and reusability, high interoperability for information exchange, flexibility on design to facilitate developers adding or removing components, and compatibility with other systems [15]. As suggested by Dagger and O’Connor [4], next generation of Learning Management Systems are envisioning scenarios where it is possible among different systems to exchange not only data but tools, functionalities, semantics, and learning workflow. Therefore, Users will be able to customize their e-Learning platforms from a diverse range of e-Learning services. To conclude our review, we would like to point that in both traditional and e-Learning environment, knowledge has to be divided in meaningful chunks of information to develop and test a particular skill. The delivery method triggers the interaction between the teacher and student. That is why we are focusing on the delivery method enhancing this interaction based on a Task Management approach.

The main goal of our research is to develop a software platform that provides facilities to integrate a Task Management issue as a core e-Learning component that can be applied to different education domains. This platform will enable clients to specify and design their Tasks hosted as a service through Cloud interfaced by Web services technology. Other e-Learning components can also be added to scale the environment capabilities, and facilitate the integration with third-party applications for high interoperability, reusability and information exchange.

The Cloud computing enables convenient, on-demand network access to a shared pool of configurable computing resources (such as networks, servers, storage, applications and services) that can be rapidly provisioned and accessed with minimal management effort or service provider interaction. Service oriented Cloud computing focuses on statelessness, low coupling, modularity and semantic interoperability [8]. The main essence of Cloud computing is to significantly virtualize the services and abstract an implementation logic from service invoker [5]. Cloud computing provides Quality of Service (QoS) guaranteed infrastructures, e.g. time, cost, reliability and hardware performance such as CPU bandwidth and memory consumption and sustains SLA oriented resource allocation [2].

As Cloud computing is rising as a technological innovation over the horizon of the computing science [2,8,11], we are undertaking a research to implement e-Learning systems in the event of the Cloud computing bolstered by a Web services technology for interface integration. The Model-View-Controller (MVC) design patterns are the main source of motivation to architect the domain of modules and interconnections. The View and Controller are represented as a Public Distributed Cloud so that both the modules to be developed independently. In the Distributed Cloud, every node is an independent Cloud that offers unique service like View Cloud offers Graphical User Interface (GUI) as a Service and Controller Cloud offers Task as a Service (TaaS). The inception of Distributed Cloud leverages the implementation of the most reliable, scalable and reusable services. The Model stores the configuration definition schema used by Controller and hence, the Model resides within a Controller Cloud. The Controller is further normalized to host a Cloud for a set of actions executing the desired output in order to outsource those actions to external Clouds and/or import external Cloud services as Application Programming Interface (API) for extending and overloading current scope of operation. Alongside this main course, we have other features required by e-Learning environment like access control by user profile, session management, rule based verification and authorization engine, alert notification engine, grading or scoring engine, pluggable problem processing engines like arithmetic or formula engine, human language engine etc. included in the proposed design.

The rest of this paper is organized as follows. In Section 2, the Task Management approach is explained along with the main features that will be generalized to create an abstract model. It includes an example that will be further developed as a prototype. Section 3 explains the architecture of Distributed Cloud for e-Learning. In Section 4, the system design issues are described including implementation details and security measures. The conclusion and future work are discussed in Section 5.

2. TASK MANAGEMENT E-LEARNING ENVIRONMENT

The original idea of Task Management study initially comes from our previous research [8] based on a comfortable task management tool to enhance collaboration between teacher and students, and applied to subjects that require complex formula verification such as mathematics and physics. The Task Management environment is implemented under client-server architecture, where clients are workstations used by teachers and students. An Exercise server application manages a database of problem collection, and a set of functions to facilitate problem preparation, distribution and automatic verification. The formula engine facilitates the problem design and can automatically create different versions of the same problem changing initial parameters input by teacher. Verification engine applies different verification strategies to evaluate not only numerical answers but also complex formula expressions applied to subjects such as mathematics, physics, statistics, chemistry, etc. which we will refer in this document as Complex Formula Expression Subjects (CFES).

2.1 Task Management Abstract Model Scheme

The main structure of the Task Management approach can be generalized in an Abstract Model (see Figure 1) that will be referred in this document as Task Management Abstract Model.
Learning support is unified under this scheme facilitating the correspondence to extend into different domains, and the integration to the e-Learning platform.

The main concepts and classes are defined as follows: **Task management** is the process to organize a set of activities or tasks that need to be executed to achieve a predetermined objective. An **Objective** is a set of tasks strongly tied together in specific order. The completion of all tasks defines the completion of the objective. A **Task** is a set of actions that need to be completed to reach the predetermined objective. An **Action** is an atomic event, which can not be further decomposed. A **Metric** is an element that collects and summarizes results to measure execution of a learning objective. This is to be used for statistical analysis and performance associated with the consumer behavior. An **Alert** is an element used for notification about the execution of each task and change of status task information. A **Provider** is the user who owns, creates or maintains all or part of the Task Management Elements. A **Consumer** is the receptor of the learning goal. It is expected to payback with specific response according to the actions consumed. An **Evaluator** is the user who does not produce or own elements. The evaluator’s participation is limited to a specific objective with a role of monitoring tasks execution and with rights to score results. **Interface** is a protocol of communication integration that individually wraps loosely coupled peers e.g. Web services. **Session** is a time window associated with a user to utilize the task.

### 2.2 Task Management Model Applied to CFES

In framework of e-Learning, the TMAM contains the basic description of the classes that will be transformed into a correspondence class of the specific domain. A prototype is to be designed applying this scheme (see Figure 2) to CFES as initial component of the e-Learning platform. The **Objective** is implemented as a **project** class with Course and Tournament as the two main variants. The objective can be extended defining new types and easily integrated under the model structure. A **Course** is defined as a set of tasks to be implemented as lectures and tests. The producer defines prerequisites for each task to trigger access to next one. It is possible to set more than one task with the same prerequisite. In this case the order of execution is optional to the consumer. The **Tournament** is the second variant of learning objective. The tasks are implemented as Exams with different levels of difficulty and a specific order of execution. A **Test** can be implemented with three different variants: An **Exercise** is an informal evaluation of the acquired knowledge. A **Quiz** is a scored test, time bounded and generated with unique values for each student. An **Exam** has the Quiz characteristics with the constraint that the time duration is less than 24 hours. A **Dashboard** class corresponds to the performance monitor implementation and presents real time scores for tournament participants, as well as specific metrics set by producer. The **Judges** are evaluator users with a lifetime in the system strictly related with the learning objectives in execution.

### 3. DISTRIBUTED CLOUD TO DELIVER TASK AS A SERVICE

#### 3.1 Architecture of Distributed Cloud

Figure 3 envisages the architectural view of the proposed Distributed Cloud based on MVC design patterns concept. The architecture concentrates on the core task processing work and therefore the MVC design patterns [6] are used in order to reuse the loosely coupled components and objects.
Figure 3. Architecture of Distributed Cloud for e-Learning

The design has identified the View and Controller as independent entities in the form of a Cloud which will communicate through Web services interface with Extensible Markup Language (XML) protocol over Simple Object Access Protocol (SOAP). The Cloud implementation is done by using Virtual Machine (VM) Cluster where a pool of infrastructure resources are load balanced by hypervisors in vertical as well as horizontal scalability. The Utility Computing is incorporated in the Controller Cloud as utility billing is charged with respect to the task's valuation as decided by the task owner and configuration of which is modeled in the database in the form of rules or templates. The end user client is going to be thin as the GUI layer is detached from the task controller in order to View as an independent Cloud which is responsible for only request and response message mapping in the Web Service Definition Language (WSDL) XML interface.

There are two modes of communication: Direct and Discovery. The Direct mode enables two Clouds to communicate through pre-shared XML Web services interface over SOAP call. In the Discovery mode, Universal Description, Discovery and Integration (UDDI), is used as a service manager wherein individual Cloud publishes their WSDL interfaces which is invoked by the client Clouds. In case of direct mode the individual Cloud has to make sure that the current updated copy of the XML interface is multicast to the associated third party Cloud. This overhead is resolved by using UDDI as interface definition publication is centralized. In the next subsection we discuss the View and Controller Clouds with respect to the task management in order to deliver TaaS.

3.2 Controller Cloud

Controller Cloud is the central point in our e-Learning task management environment that eventually delivers TaaS.

Controller Cloud has two controllers viz. Session Controller and Task Controller. Task Controller is exclusively identified as a Cloud within the main Controller Cloud to expose TaaS. The Task Controller Cloud communicates in XML messages. The XML response rendered from the Task Controller Cloud can be represented in any format at the user end. Consequently, the services of the Task Controller Cloud are exposed in both Business to Consumer (B2C) and Business to Business (B2B) modes of interaction. A Model is locally used by Task controller Cloud to configure the schema definition of the data and task processing logic rules as well as templates.

3.2.1 Session Controller

The Session controller evaluates and verifies the session of every task request with respect to the user session configuration as well as binds the request parameters from View to the corresponding task service definition's Data Transfer Object (DTO). The Session controller is a first step whenever user and task authentication is required prior to the service invocation. Session controller is not mandatory in case only TaaS is needed by an External Cloud. The session controller performs following tasks:

**User ID Authentication:** As e-Learning applications are vulnerable for re-purposing and other network attacks, the session controller authenticates the user by using a two level authentication scheme. If first level authentication (e.g. user id and user specific credential like date of birth) fails, the user account is soft-locked (i.e. user account is locked only for short period of time) and if second level authentication (e.g. user's pin or password) fails, the user account is hard-locked (i.e. user account is permanently locked).

**User Session Authorization:** If the valid user successfully logs into the e-Learning application, a session ID is assigned to the...
user by Session Controller. This session ID is verified (by session controller) for all subsequent communication between View and Controller for the given user.

**Task Request Binding or Mapping:** If the above first two authentications are successful, Task Request Binder maps the user’s request parameters (e.g. input parameters in HTTP request) to the DTO parameters of Task Controller interface, as given in Figure 4. This mapping is called as View and Controller Task Request Binding (VCTRB). Hence, this is a message passing from View to Controller. This binding isolates the Task Controller from the GUI modifications done at View and therefore, View as well as Controller becomes the loosely coupled modules.

### 3.2.2 Task Controller Cloud

In order to observe TaaS, the logic associated with it is wrapped inside a Web services XML interface in order to reuse export and import of the task objects from External Clouds. As this Cloud is identified inside the main Controller Cloud, it is invoked by the Session controller in the Direct mode. The Task Controller Cloud manages the task objects in a container across the VM cluster. Task object management is done by using many components like Transaction Manager (to control the 2-phase commit XA connection across the distributed system needed by the Task Controller), Task Object Factory (to control the object generation during coordination among actions), Task Object Pool (to control object store for significant performance boosting), Task Object Observer (to control the change in object state and notification) and Task Object Monitor (to control concurrent usage of the same object for mutual exclusion) [6]. The main components associated with the Task Controller Cloud are as following:

**Task Request and Response Parser** verifies and parses the request received from the VCTRB for the operation of the task controller. The data value is unmarshalled or deserialized (converting back to DTO) for the specific task processor [3]. This parser also serializes or marshals the Controller’s response which is returned to View after task execution.

**Verification Engine** performs the answer verification (discussed in more details in section 4.1) and validation of the given task request with respect to the request fields, parameter's data type and width, request specific validations etc. The verification processor is a configurable engine which is defined by developing certain rules and using those rules as a template (see Table 1). The rule or template designed is stored in the Model.

**Authorization Engine** is responsible for authorization of a request associated with the usage of the task by task consumer and task owner e.g. If a participant (task consumer) of the tournament is submitting his answer, this request is submitted for the authorization of a teacher (task owner) or the group of teachers. The authorization processor is a configurable engine which is configured by developing rules and templates specific to the user and/or task. The rules can also be developed according to the hierarchy of the users (authorizer). These rules and templates are stored in the Model or task database (see Table 2).

**Alert Engine** triggers the notification services whenever the state of the task is changed. The task owner as well as consumer is notified for the task state changes by Short Message Service (SMS) or E-Mail channels. This is a background asynchronous process running in the task object container in the form of a timer thread e.g. after the grade registration by the teacher, the task status is changed to complete and corresponding notification is sent to student.

**Grading or Scoring Engine** keeps track of the student's progress report by evaluating the transcript as per the task rule definition devised by the task owner like teacher. While defining or publishing a task like Quiz, the teacher has to configure the grading rule and policy for that task. The student subscribes or registers for the task like Quiz and student's performance in terms of grade or score is computed by the Grading engine. If the authorization rule is enabled for this task, the Grading engine waits for the final authorization otherwise by default Verification

---

**Figure 4. VCTRB Sample XML protocol structure for message passing from View to Controller**

**Table 1. Verification Engine: Rule based sample Schema**

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Data Value</th>
<th>Data Length</th>
<th>Data Type</th>
<th>Valid Characters</th>
<th>Task Specific Validation (supplementary macro program is required)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tournament</td>
<td>participant</td>
<td>50</td>
<td>alphabet</td>
<td>e.g. participant can not be the owner of the task like teacher</td>
<td></td>
</tr>
<tr>
<td>tournament</td>
<td>question</td>
<td>1000</td>
<td>alphanumeric</td>
<td>e.g. %*</td>
<td></td>
</tr>
<tr>
<td>tournament</td>
<td>answer</td>
<td>20</td>
<td>number</td>
<td>e.g. answer can not be a negative integer</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2. Authorization Engine: Rule based sample Schema**

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Task Consumer</th>
<th>Task Owner</th>
<th>Task Authorizer 01</th>
<th>Task Authorizer 02</th>
<th>Task Specific Authorization (supplementary macro program is required)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tournament</td>
<td>student</td>
<td>professor</td>
<td>lecturer</td>
<td>professor</td>
<td>e.g. answer can not be submitted after the given deadline</td>
</tr>
</tbody>
</table>
engine validates the task and according to the result of validation, the task gets authorized for either failure or success.

3.2.3 Supporting Engines
The Task Controller Cloud implements supporting engines to abstract certain task processing related functionalities. Such abstraction enables the reusability and scalability of the task operation like mathematical query processing task, language related query processing task etc. The supporting engines are plugged as add-ons and therefore, any type of task processing can be hooked to the main controller. As these engines are loosely coupled with the main controller, any third party Cloud utility can be interfaced as an add-on to the main controller. The task controller can use following engines for data processing:

**Arithmetic and Formula Engine** provides certain helper operators to perform mathematical operations like integration, derivation, limits, permutation, addition, subtraction etc.

**Human Language Engine** processes the grammar of a specific human language and thereby solves language related problems, for example, in English language, problems related to the active/passive voice, direct/indirect sentences, vocabulary etc.

**Logical Reasoning Engine** processes logic related tasks of deduction, induction, abduction e.g. finding an odd number from a given series of numbers etc.

3.2.4 Model
The Model is a local database schema used by the task controller. The Model stores the schema definition for the task including user profile, access rights, user relationship like student-teacher association, problem definition, session control, verification templates, authorization templates, task rules for grading, alert notification etc. The Model is preferably a relational database because the various entities associated with Controller are defined in a normalized relationship model. The database connection between Task Controller and database Model is implemented by either Open Database Connectivity (ODBC) bridge in the form of a database driver or Object-relational Mapping (ORM).

3.3 View Cloud
The View is a GUI Cloud to construct the presentation look and feel design and to exchange input and output messages along with the end client user. The GUI is outsourced from the core task processing activity and therefore, the View is built autonomously as per the client requirements. Hence, an identical response message from Controller Cloud is realized in different views at the user end. Likewise, the View can support a wide range of clients including personal computer, mobile phone, Personal Digital Assistant (PDA), Tablet etc by hosting a website or portal. The client can access this interface over Ubiquitous channels like the Broadband Internet, Wireless standards like Bluetooth, ZigBee, Infrared, Wi-Fi etc. Such loose coupling with the task processing logic significantly decreases the development, debugging and troubleshooting efforts as well as time.

Web container is clustered to host the static/dynamic HyperText Markup Language (HTML) pages, Servlet, Cascaded Style Sheets (CSS), JavaScript code files, images, movies, flash objects, and applets etc. to construct the presentation at client end. The Extensible Stylesheet Language (XSL) transformation technique is used to generate the HTML output by imposing the XML response message as received from Controller Cloud onto the XSL files and applying the XML Schema Definition (XSD) in order to validate the patterns of the XML response data.

3.4 External e-Learning Cloud
The proposed TaaS Cloud model is scalable in terms of processing the logic. Task Controller Cloud has Web services interface to enhance the task execution intelligence by importing third party Cloud services and at the same time by hosting the task for third party Cloud for exporting TaaS. Therefore, the external e-Learning Cloud includes Moodle Course Management System (CMS), External Task Processing Engines or Processors, Programming Contest Control Project, External user profile model hosted as Data as a Service (DaaS) etc. The External e-Learning Cloud can only communicate with the Task Controller Cloud either in the Direct mode or in the UDDI mode.

4. DESIGN ISSUES
Complexities associated with the e-Learning environment for rendering knowledge based online training services, needs to chalk out rationally. Accordingly, let us discuss few common issues related to the e-Learning environment with respect to the proposed research work.

4.1 Answer Verification Strategies
There are three main strategies applied for automatic verification. This section contains a summary and more detail information can be found in [13].

**Numerical Answer Verification.** To evaluate numerical answers, the verification engine uses the initial values and a precision rate that teacher defines when the problem is created. It simply compares the numerical answers of the student and teacher. In case result is different, it calculates the absolute value of the difference and performs a comparison with the precision rate, storing the result for teacher review.

**Multiple Answer Verification.** This strategy is applied when the evaluation is performed for different variants of the same problem. In this case teacher’s answer is stored as a formula. The engine calculates teacher’s initial parameters using the stored formula and applies numerical answer verification to evaluate student’s answer comparing with values generated by teacher.

**Formula Answer Verification.** This strategy is applied when student’s answer is an equation system (see Figure 5). Precision rate and formula are stored in teacher’s Test Data. In this case the formula verifier composes the student and teacher formula, a symbolic transformer is applied to simplify the composed expression. The absolute value of the difference is compared with the precision rate. If the difference is within the precision rate the answer is scored as correct, otherwise is scored incorrect. In both cases the value is stored for teacher review.

4.2 Task Management GUI
The GUI for Task Management environment is a Web interface that can be supported under any browser. It also includes a simple interface for PDA that has no embedded Java component.
Teacher Interface. Workstation Teacher Interface provides tools for problem design (see Figure 6). Teacher can use a formula editor to customize the equation variables and operators, and hence offers Multilanguage support. During problem preparation, the teacher can verify the input data and correct it if necessary.

Student Interface. The student’s GUI interface (see Figure 7) is generated in correspondence with the problem definition. Formula Construction Palette shows variables and operators related to the particular problem is being resolved. If defined by teacher, each problem might include hint information to show more explanation to resolve the problem. The result information is automatically displayed after answer has been submitted by student.

The prototype integrates the formula and verification engines, which are hosted under the Task Controller Cloud. The GUI is designed based on the View Cloud to support a wide range of clients including workstations, mobile phones, PDA, Tablet PCs etc.

4.3 Security in Cloud
The analysis of the latest market data published, such as the IDC [14] shows that the security is a prime hindrance to the advancement and adoption of the Cloud computing driven economy due to it’s vulnerability to various attacks caused by unauthorized intrusion, malicious computing, human errors, privacy, natural disaster etc. The risk associated with security fulfillment should be jointly borne by both service provider and service consumer for speedy and hassle free Cloud implementation because the public channels of communication like the Internet is not under the control of a single autonomous body and therefore it is unsafe. This proposes a security technique which is defined by the consumer of service and implemented by the provider of service. On the other hand, such security technique should be flexible enough for customizations as per the client’s desired level of security on a case by case basis. Therefore, we propose to devise a security kernel wrapper [1] to abstract the core Cloud processing engine, for example a hypervisor in VM cluster, by forming a protective coating around Cloud engine. The concept of security kernel is inherited from the operating system kernel. The security kernel is designed for the basic standard network security measures; it is provisioned to monitor the flow of data and events of various security operations according to the user defined customizations and rules. The kernel acts as a pluggable security view to the core Cloud processing engine and provides a shield against the network based attacks. This idea induces a high esteem into the accreditation of the e-Learning environment by educational institutions over non-trusted internet channel.

5. CONCLUSION AND FUTURE WORK
In a nutshell, the proposed research has the potential to be realized as a Software Development Platform of Distributed Cloud Computing for implementing Task Management inside the e-Learning environment with a great ease of development and maintenance. The common issues related to the Quality of Service (QoS) such as scalability, performance, and security, can be handled depending on the desired level of QoS. This alleviates the bottleneck of systems adoption by various kinds of vendors as the risk associated with the QoS is born by both service provider and consumer.
Cloud. Thereby, the presentation logic has been outsourced, and the developers can be concentrated on computational services related to e-Learning subject itself. The B2B and B2C interaction modes leverage an endpoint usability of the e-Learning computational services that are deployed in the Task Controller Cloud.

In the proposed model, the Task Controller Cloud delivers the TaaS. The controller can be enhanced or scaled in order to tackle any kind of tasks associated with any field of study like human language exercise, mathematical problems, logical and critical reasoning, statistical problems, etc. The same is also true for the View Cloud because it delivers the GUI as a Service. The comparison with Web services implementation approaches specified in [15], our approach adds significant improvement in open architecture, durability, reusability, high interoperability for information exchange, flexibility and compatibility. This is reached because of the loosely coupled View and Controller. The proposed Cloud Security Kernel is provisioned for implementation of the customized security solutions requested by the client. The risk associated with the unsafe communication channel is jointly borne by both provider and consumer of the Cloud services and thereby easing the sound formulation of the Service Level Agreement (SLA) to accelerate the Cloud driver e-Learning business.

Going forward, we are planning to introduce the JavaScript Object Notation (JSON) for representing internal data model of the task rather than using XML [12]. The performance economy associated with the XML data model is inversely proportional to the size of the XML data and hence, the parsing of the bigger size XML becomes cumbersome. The JSON standard is a most lightweight standard to model the data representation.

The Cloud service prototype was built on the top of the VM cluster computing, and we are planning to implement the grid computing to get the maximum possible utilization of the underlying resources by harnessing together many servers running in parallel. A change from VM cluster to grid is only related to the performance enhancement and does not specifically influences the task management process.

6. ACKNOWLEDGMENTS

This work is supported by the Japanese Society for the Promotion of Science, Grant-in-Aid for Scientific Research: 21500948, 2009-2011.

7. REFERENCES


