PENERAPAN ARSITEKTUR *MODEL VIEW CONTROLLER (MVC)* DALAM RANCANG BANGUN SISTEM KUIS *ONLINE* ADAPTIF

Arief Hidayat[1], Bayu Surarso[2]
[1]*Program Studi Sistem Informasi, STMIK ProVisi Semarang*
Jl. Kyai Saleh no 12-14 Semarang 50243
Telp. (024) 8418206
[2]*Program Studi Ilmu Komputer, Universitas Diponegoro*
Jl. Prof Sudharto SH, Tembalang, Semarang 50147
E-mail: rifmillenia@gmail.com, bayus@undip.ac.id

ABSTRAKS
Sistem kuis online adaptif menjadi lebih bersifat pribadi karena model pertanyaan yang disajikan secara khusus dirancang bagi siswa sesuai dengan tingkat kemampuan mereka. Siswa akan lebih mengenal kekuatan dan kelemahan dalam proses belajar mereka karena mereka tidak akan menuju ke tingkat kesulitan yang lebih tinggi jika mereka tidak memenuhi nilai yang dipersyaratkan pada tingkat tertentu. Rancang bangun sistem ini akan lebih mudah dikembangkan jika menggunakan pola desain berarsitektur model view controller (MVC). Arsitektur ini membagi aplikasi menjadi tiga bagian secara konsep yang terpisah yaitu model, view, dan controller, masing-masing dapat dikembangkan secara terpisah antara satu dengan yang lainnya, sehingga perubahan pada satu bagian memiliki dampak minimal pada bagian lain. Bagian model digunakan untuk mendefinisikan suatu cara dimana data dapat diakses, bagian view menghasilkan keluaran jika diberikan data, dan bagian controller menerima perintah dan mengatur aplikasi untuk tugas dan tampilan yang sesuai. Hasil dari rancang bangun ini adalah sebuah sistem penilaian siswa berdasarkan kemampuan, pengetahuan dan pilihan dari masing-masing siswa secara online.

Kata Kunci: MVC (model view controller), kuis, online, adaptif

1. PENDAHULUAN

Suatu perangkat lunak agar lebih berguna, tentu saja harus berinteraksi dengan sesuatu, kadang-kadang berinteraksi dengan perangkat lain, dan yang lebih sering dengan manusia. Perangkat lunak memerlukan beberapa antarmuka untuk membuat interaksi lebih efisien. Beberapa aplikasi berbasis web membutuhkan lebih banyak sumber daya untuk membangun sebuah antarmuka yang efektif dan terstruktur secara modular dan itu lebih baik dari pada hanya sekedar logika proses (Deacon, 2009).

Sistem berbasis web konvensional, masih mencampur kode program antara logika proses dan antarmuka. Antarmuka dalam sistem berbasis web konvensional hanya dapat digunakan dalam satu proses logika, hal ini akan mengurangi modularitas aplikasi, membuat pemeliharaan sistem lebih sulit, dan juga membuat antarmuka sulit untuk dimodifikasi ketika akan digunakan untuk aplikasi lain (Deacon, 2009). Misalnya, script server (PHP, JSP, ASP, dan lain-lain) masih dicampur dengan script tampilan (HTML, WML, JavaScript, dan lain-lain).

Permasalahan tersebut telah menimbulkan gagasan untuk memisahkan logika aplikasi dengan antarmuka, sehingga aplikasi yang dibangun dengan mudah dapat diganti antarmukanya setiap saat. Pada tahun 1970-an penemu *smalltalk, Trygve Reenskaug,* mendefinisikan sebuah arsitektur untuk menyelesaikan masalah tersebut, yang disebut arsitektur *model-view-controller* (MVC) yang memisahkan logika bisnis dan presentasi (tampilan) aplikasi.

Sistem berbasis web telah banyak diterapkan di berbagai bidang, khususnya bagi yang membutuhkan suatu sistem yang bersifat *online* dengan segala kelebihannya, termasuk bidang pendidikan. Institusi pendidikan banyak yang memanfaatkan sistem berbasis web, dari mulai sekedar menampilkan berita atau informasi, pendaftaran *online*, pembelajaran *online*, sampai pada penilaian online.

Penilaian sering digunakan untuk mengukur performa peserta didik (Quinn dan Reid, 2003), Alotaiby & Chen (2005) menggambarkan penilaian sebagai salah satu komponen utama yang membantu peserta didik dalam belajar. Kuis, yang berfungsi sebagai jenis penilaian (QuestionMark dan League, 2004) adalah yang paling banyak digunakan dan merupakan metode penilaian yang dikembangkan dengan baik dalam pendidikan tinggi (Brusilovsky dan Miller, 1999).

Kuis online memberikan banyak keuntungan, salah satunya yaitu memungkinkan peserta didik untuk mengerjakan kuis kapan saja. Penggunaan web sebagai sarana untuk mengerjakan kuis tidak mengatasi isu penting tentang peningkatan proses belajar peserta didik, akan tetapi hanya sebagai media teknologi revolusi kuis. Salah satu permasalahan utama dengan kuis *online*, adalah kurangnya nilai signifikan yang disebut 'Personalisasi'. Tanpa personalisasi, sistem

memperlakukan semua peserta didik dengan cara yang sama. Personalisasi dibutuhkan sistem untuk menyesuaikan dengan kebutuhan peserta didik secara otomatis yang disebut 'Adaptif' (Santally dan Senteni, 2005). Kemampuan sistem adaptif dapat memutuskan pilihan mana yang terbaik bagi pengguna berdasarkan model pengguna mereka. Sistem adaptif terus melacak aktivitas pola pengguna dan mencoba menyesuaikan antarmuka atau konten yang cocok untuk pengguna yang berbeda keahlian, pengetahuan dan preferensi yang berbeda (Kules, 2000).

Istilah adaptif sering membingungkan dengan istilah beradaptasi (*adaptable*). Beradaptasi memungkinkan pengguna untuk mengontrol penyesuaian (Kules, 2000) yang memungkinkan pengguna untuk menyesuaikan sistem sesuai dengan kebutuhan sendiri (Cheng dan Kinshuk, 2004). Adaptif sendiri adalah fitur yang digunakan untuk mengukur tingkat peserta didik saat ini dari domain kompetensi (Cheng dan Kinshuk, 2004). Sistem ini disebut adaptif jika dapat mengubah atribut sendiri secara otomatis sesuai dengan kebutuhan pengguna (Weibelzahl, 2002).

Arsitektur *model view controller* (MVC) akan diterapkan dalam rancang bangun sistem kuis online adaptif ini untuk mempermudah dalam pengelolaan dan pemeliharaan sistem, karena adanya pemisahan antara *model*, *view* dan *controller*.

2. TINJAUAN PUSTAKA

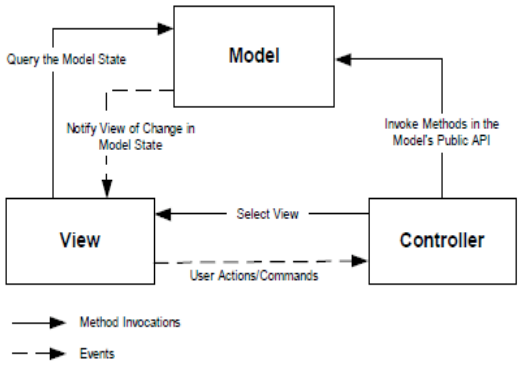
2.1 Arsitektur MVC (Model-View-Controller)

Model-View-Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu *Smalltalk* (*Trygve Reenskaug*) untuk meng-*enkapsulasi* data bersama dengan pemrosesan (*model*), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk direpresentasikan pada sebuah *user interface* (Deacon, 2009). MVC mengikuti pendekatan yang paling umum dari *Layering*. *Layering* hanyalah sebuah logika yang membagi kode kita ke dalam fungsi di kelas yang berbeda. Pendekatan ini mudah dikenal dan yang paling banyak diterima. Keuntungan utama dalam pendekatan ini adalah penggunaan ulang (*reusability*) kode (Satish, 2004).

Definisi teknis dari arsitektur MVC dibagi menjadi tiga lapisan (Burbeck, 1992)

a. *Model*, digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi yang berhubungan dengan pemrosesan data. Sebuah model meringkas lebih dari sekedar data dan fungsi yang beroperasi di dalamnya. Pendekatan model yang digunakan untuk komputer model atau abstraksi dari beberapa proses dunia nyata. Hal ini tidak hanya menangkap keadaan proses atau sistem, tetapi bagaimana sistem bekerja. Sebagai contoh, programmer dapat menentukan model yang menjembatani komputasi *back-end* dengan *front-end* GUI (*graphical user interface*).
b. *View*, bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. *View* biasanya memiliki hubungan 1-1 dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. *View* melekat pada *model* dan me-*render* isinya ke permukaan layar. Selain itu, ketika *model* berubah, *view* secara otomatis menggambar ulang bagian layar yang terkena perubahan untuk menunjukkan perubahan tersebut. Terdapat kemungkinan beberapa *view* pada *model* yang sama dan masing-masing *view* tersebut dapat me-*render* isi *model* untuk permukaan tampilan yang berbeda.
c. *Controller*, menerima input dari pengguna dan menginstruksikan *model* dan *view* untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, *controller* bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap respon aplikasi. Sebagai contoh, ketika pengguna mengklik tombol atau memilih item menu, *controller* bertanggung jawab untuk menentukan bagaimana aplikasi seharusnya merespon.

Model, *view* dan *controller* sangat erat terkait, oleh karena itu, mereka harus merujuk satu sama lain. Gambar 1. mengilustrasikan hubungan dasar *Model-View-Controller*.



Gambar 1. Hubungan antara model, view, dan controller (Gulzar, 2002)

Arsitektur MVC memiliki manfaat yaitu pemisahan antara *model* dan *view* memungkinkan beberapa *view* menggunakan *model* yang sama. Akibatnya, komponen *model* sebuah aplikasi lebih mudah untuk diterapkan, diuji, dan dipelihara, karena semua akses ke *model* berjalan melalui komponen ini (Balani, 2002)..

2.2 Kuis Adaptif

Pertanyaan dalam kuis dapat disajikan sesuai dengan tingkat kemampuan peserta didik, hal ini dapat dilakukan dengan menambahkan fitur adaptif di kuis,. Perkiraan performa peserta didik didasarkan pada dua faktor (Gouli et al., 2001) yaitu:

a. nilai dari jawaban benar yang diberikan oleh

peserta didik, dan
b. tingkat kesulitan dari pertanyaan-pertanyaan yang mampu dijawab benar oleh peserta didik.

Gouli et al. (2001) telah menunjukkan tiga pertanyaan penting ketika mengembangkan kuis adaptif yaitu:
a. pertanyaan mana yang akan diberikan kepada peserta didik ?
b. bagaimana mengukur perkiraan kemampuan para peserta didik dan juga bagaimana mempertahankan atau memperbarui tingkat kemampuan peserta didik?
c. apa kriteria untuk menghentikan kuis ?

Berdasarkan pertanyaan di atas, Gouli et al. (2001) telah menggambarkan tentang bagaimana prosedur penilaian adaptif bekerja pada umumnya.
a. Tahap awal sebuah pertanyaan dengan tingkat kesulitan beragam disajikan, karena pada tahap ini, sistem tidak tahu tingkat kemampuan peserta didik. Sistem ini mengasumsikan bahwa peserta didik memiliki tingkat pengetahuan beragam untuk suatu pokok bahasan.
b. Sistem akan melakukan pemeriksaan setelah peserta didik menjawab dan mengirim jawaban,. Sistem akan memperbarui kemampuan peserta didik menjadi lebih tinggi jika jawabannya benar, jika tidak, sistem akan mengatur ke tingkat yang lebih rendah.
c. Sistem kemudian memilih pertanyaan berikutnya yang sesuai untuk peserta didik menurut tingkat kemampuan mereka.
d. Proses ini berlanjut sampai satu kriteria terakhir tercapai seperti peserta didik keluar dari sistem atau peserta didik tersebut telah mencapai pada tingkat tertinggi dari suatu subjek.

3. METODOLOGI PENELITIAN

Rancang bangun sistem kuis online adaptif berarsitektur *model-view-controller (*MVC*)* ini menerapkan metodologi SDLC (System Development Life Cycle) dengan menggunakan pendekatan berorientasi objek. Pengembangan yang dilakukan yaitu *iterative* dan *incremental*. Tahapan-tahapan penelitian adalah sebagai berikut:
a. Pengumpulan kebutuhan
   Tahap ini mendefinisikan kebutuhan sistem apa yang harus dilakukan atau bagaimana seharusnya melakukan. Ada dua jenis kebutuhan yang akan diidentifikasi. Jenis pertama adalah kebutuhan fungsional yang menangkap apa yang sistem lakukan. Jenis kedua adalah kebutuhan non-fungsional yang menangkap kendala pada sistem itu sendiri, serta kendala yang dihadapi sistem dalam tahap pengembangan.
b. Analisa sistem
   Pada tahap analisa sistem, masing-masing kebutuhan fungsional dan non-fungsional dianalisa secara rinci. Teknik analisa berorientasi objek telah dipilih untuk mengidentifikasi lebih lanjut kebutuhan sistem yang di rinci dengan menggunakan UML (Unified Modeling Language). Kebutuhan sistem dijabarkan dalam bentuk notasi grafis dengan menggunakan UML. Setiap use case dikaitkan dengan aktor, sehingga setiap use case menyatakan tujuan aktor dalam menggunakan sistem Hal ini juga menunjukkan interaksi antara aktor dan sistem.
c. Desain sistem
   Pada tahap desain sistem, desain teknis mulai didefinisikan yang berisi desain terinci untuk masing-masing use case. Semua use case yang disajikan dalam tahap analisa akan diperinci dan disajikan dalam tiga diagram yaitu *sequence diagram*, *class diagram*, dan *diagram aktifitas*. *Sequence diagram* digunakan untuk menggambarkan interaksi yang diatur dalam urutan waktu. *Class diagram* digunakan untuk memodelkan class-class yang akan digunakan, yang nantinya akan juga ditransformasikan dalam bentuk tabel-tabel suatu database sebagai tempat penyimpanan data, sedangkan *diagram aktifitas* untuk memodelkan proses yang berlangsung dalam sistem kuis online adaptif ini. Desain database digambarkan sebagai bentuk pemodelan data dan desain user interface juga akan digambarkan dalam tahapan ini sebagai bentuk antar muka sistem dengan pengguna.
d. Implementasi
   Implementasi dengan menggunakan arsitektur MVC yaitu akan membuat *data model* untuk merepresentasikan informasi dari database, *view* untuk menampilkan data, dan *controller* yang akan menggabungkan keduanya bersama-sama dan menangani tugas lain.
e. Pengujian
   Pengujian sistem kuis online adaptif ini dilakukan dengan tujuan untuk mengetahui adanya kesesuaian antara fungsi-fungsi atau layanan-layanan sistem yang diimplementasikan dengan hasil analisa kebutuhan yang sudah ditentukan pada tahap analisa kebutuhan fungsional maupun non-fungsional. Pengujian juga dilakukan untuk mengetahui apakah sistem tersebut dapat diakses menggunakan beberapa software internet browser. Selain itu dengan dilakukannya pengujian akan dapat diketahui adanya kesalahan- kesalahan dalam proses *coding* maupun dalam menghasilkan output program sehingga kesalahan-kesalahan tersebut dapat diperbaiki. Metode pengujian yang dilakukan adalah *white box* dan *black box*.

4. IMPLEMENTASI ARSITEKTUR MVC

Arsitektur MVC diimplementasikan dalam setiap modul aplikasi dalam sistem kuis online adaptif ini. Setiap modul memiliki satu *model*, satu *controller*, dan beberapa *view*. Berikut ini akan dijelaskan implementasi arsitektur MVC pada component kuis.

4.1 *Model*

Model perlu dibuat sebelum mulai memisahkan *view* untuk mendapatkan informasi yang akan kita tampilkan. Bagian *front-end* digunakan untuk siswa dan pengajar, sedangkan bagian *back-end* digunakan untuk administrator. Berikut ini adalah penggalan *script* untuk menampilkan data kuis secara tunggal dari bagian *front-end* seperti yang ditunjukkan dalam Gambar 2.

```
<?php

defined('_JEXEC') or die('REstricted Access');

jimport('joomla.application.component.model');

class KuisModelKuis extends JModel
{

        function __construct()
        {
        parent::__construct();

        $path = JPATH_COMPONENT.DS.'tables';
                $this->addTablePath($path);
        }

        function getKuis($id)
        {
        $db = $this->getDBO();
        $table = $db->nameQuote('#__tbl_kuis');
        $key = $db->nameQuote('id');
        $query = "SELECT * FROM ".$table."
                  WHERE ".$key." = ".$id;

        $db->setQuery($query);
        $kuis = $db->loadObject();

        if($kuis == null) {
        JError::raiseError(500, 'Kuis ['.$id.']
       not found.');
        }else{
                return $kuis;
        }
        }
}
   ?>
```

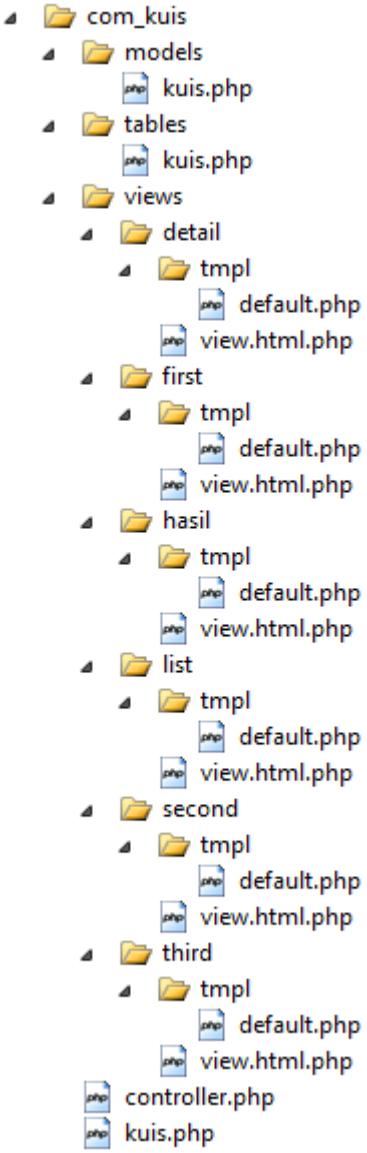
Gambar 2. Script kode *model* untuk data kuis secara tunggal

Pertama, mengimpor library model data *Joomla*, kemudian mendeklarasikan *class model* dengan nama KuisModelKuis sebagai ekstensi dari *Jmodel*. Nama *class model* harus terdiri dari nama komponen lalu diikuti oleh kata *Model* dan nama file *model* Fungsi member tunggal getKuis() mengecek untuk melihat apakah data kuis telah di-*load* berdasarkan id-nya, di dalamnya terdapat perintah untuk melakukan *query* terhadap database. Simbol prefix #__ pada awal nama tabel untuk menjamin penggunaan prefix yang benar. Metode *nameQuote()* untuk mengenkapsulasi nama dari elemen query yang menjamin bahwa nama elemen dienkapsulasi dalam *delimeter* yang benar. Hasil query di-*load* dengan menggunakan metode *loadObject()*.

4.2 *View*

Proses selanjutnya setelah *model* dibuat, yaitu diperlukan beberapa kode yang akan menampilkan informasi tersebut. Sejauh ini, file dengan akhiran. html.php telah berfungsi dengan baik untuk melakukan tugas ini. Namun, desain yang sudah ada agak kaku, *class* output HTML disertakan dan layar yang inginkan dipanggil untuk ditampilkan. Melalui penggunaan *view*, dapat membuka layar tersebut sampai dengan tingkat admin sebagai pilihan.

Keseluruhan output dalam file tunggal mulai ditinggalkan, sebagai gantinya dibuat *folder* secara terpisah untuk masing-masing *view*. *Folder* tersebut akan memuat *sub folder* untuk beberapa jenis *view* yang akan ditampilkan, dan masing-masing *sub folder* tersebut terdapat *sub folder tmpl* untuk *template*, seperti yang ditunjukkan dalam Gambar 3.



Gambar 3. Hirarki component kuis

Setiap *view* dapat menyertakan beberapa *template*, akan tetapi perlu sebuah objek *view* untuk

mengelola template tersebut yaitu sebuah file *view.html.php*, sebagai contoh file *view.html.php* dibawah *folder component/com_kuis/views/list* seperti yang ditunjukkan dalam Gambar 4.

```
<?php

defined('_JEXEC') or die('Restricted Access');

jimport('joomla.application.component.view');

class KuisViewList extends JView
{
function display($tpl = null)
   {
       //Add file css
    $document =& JFactory::getDocument();
    $document>addStyleSheet
    ('components'.DS.'com_bab'.DS.'css'.
     DS.'style.css');

     $session =& JFactory::getSession();

       //Get kuis from the model
     $model =& $this->getModel('kuis');
     $kuiss = $model->getKuiss($session->get
     ('usr_id'),JRequest::getVar('field'),
     JRequest::getVar('sort'));

     $this->assignRef('kuiss',$kuiss);

     parent::display($tpl);
    }
}
   ?>
```

Gambar 4. Script kode *view* untuk menampilkan data kuis tunggal

Nama dari class view harus terdiri dari nama komponen lalu diikuti oleh kata *View*, dan nama folder *view*. *View* dibuat dengan mewarisi *view* dari *class Jview*, untuk itu harus mengimpor *class*-nya. Alur dari sebuah *view* adalah pertama, mengambil *model* yang namanya sesuai dengan *controller* yaitu model *kuis*, kemudian mengeksekusi metode *getKuis()* dalam *model* kuis dan memberikan hasilnya ke variabel *$kuis*, setelah itu memberikan variabel kuis yang berisi *$kuis* ke layout, dan terakhir mengeksekusi metode *display* dari *class Jview* yang akan mengeksekusi *template* berdasarkan isi dari variabel *$tpl* dan jika isinya kosong (*null*) maka metode ini akan mengeksekusi file *default.php* yang terdapat dalam direktori *tmpl*.

4.3 *Controller*

Controller bekerja berdasarkan *task* apa yang telah diminta dan berdasarkan *task* tersebut maka *controller* mengambil data dari *model* dan mengirimkan data dari *model* tersebut ke *view*. Jadi tugas *controller* adalah bekerja berdasarkan inputan *user* yang kemudian dikenal dengan nama *task*, memanggil metode pada *model* untuk memanipulasi data pada tabel, dan mengirimkan data dari *model* ke view untuk ditampilkan pada *browser*. Gambar 5 menunjukkan sebagian *script* kode sebuah *controller* kuis yang akan mengendalikan alur program .

```
<?php

defined('_JEXEC') or die('Restricted Access');

jimport('joomla.application.component.controller');

class KuisController extends JController
{
        function __construct()
        {
                parent::__construct();

                $path = JPATH_COMPONENT.DS.'models';
                $this->addModelPath($path);

                $path = JPATH_COMPONENT.DS.'views';
                $this->addViewPath($path);
        }

        function display()
    {
        $this->cek_login_pengajar();

        $view =& $this->getView('list','html');
        $model =& $this->getModel('kuis');
        $view->setModel($model,true);

        $view->display();
        }

}

   ?>
```

Gambar 5. Script kode sebuah *controller*

Nama dari *class controller* harus terdiri dari nama komponen lalu diikuti oleh kata *Controller*. *Controller* dibuat dengan mewarisi *model* dari *class Jcontroller*, untuk itu harus mengimpor *class*-nya. Fungsi *display()* dipanggil secara *default* ketika *controller* dieksekusi, dengan alur yaitu pertama, mendapatkan sebuah objek *view* dengan melewatkan nama *view*, dan menuliskan ke dalam fungsi member *getView()*, kemudian memilih *model* untuk *view* dengan metode *getModel()* dan mengaturnya dalam *view* dengan metode *setModel()*, terakhir menampilkan *layout view* yang diinginkan.

5. PENGUJIAN DAN PEMBAHASAN

5.1 Pengujian Sistem

Pengujian sistem kuis online adaptif ini dilakukan dengan tujuan untuk mengetahui adanya kesesuaian antara fungsi-fungsi atau layanan-layanan sistem yang diimplementasikan dengan hasil analisa kebutuhan yang sudah ditentukan pada tahap analisa kebutuhan fungsional maupun non-fungsional. Pengujian juga dilakukan untuk mengetahui apakah sistem tersebut dapat diakses menggunakan beberapa software internet browser. Selain itu dengan dilakukannya pengujian akan dapat diketahui adanya kesalahan- kesalahan dalam proses *coding* maupun dalam menghasilkan output program sehingga kesalahan-kesalahan tersebut

dapat diperbaiki.

Metode pengujian yang dilakukan adalah *white box* dan *black box*. Pengujian *white box* adalah pengujian yang dilakukan pada internal sistem dan difokuskan pada penemuan kesalahan struktur kode-kode program atau logika pemrograman selama perangkat lunak dibangun. Pengujian *white box* tidak akan dijelaskan karena pengujian tersebut telah dilakukan bersamaan dengan proses pembangunan sistem.

Pengujian *black box* adalah pengujian yang difokuskan pada persyaratan fungsional atau kebenaran input dan output yang dihasilkan dari perangkat lunak yang dibangun. Pengujian *black box* ini akan dilakukan dengan cara memberi input dari pengguna kepada sistem yang sudah berjalan dan mengamati hasil output dari sistem. Pengujian tersebut akan dilakukan pada setiap use case untuk mengetahui kesesuaian fungsi dari sistem. Prosedur pengujian yang dilakukan pada proses pengujian terhadap sistem kuis online adaptif adalah seperti terlihat pada bagian di bawah ini:

a. Menentukan data-data yang akan digunakan untuk keperluan pengujian perangkat lunak.
b. Menentukan metode pengujian dan kriteria evaluasi hasil pengujian untuk masing-masing use case yang ada di dalam sistem.
c. Melakukan pengujian untuk masing-masing use case seperti yang tertera dalam Tabel 1 dengan menggunakan data yang sudah disiapkan sebelumnya dan membandingkan hasilnya dengan kriteria hasil pengujian.

Tabel 1. Daftar use case, metode pengujian, dan kriteria evaluasi hasil pengujian

Id. Kasus Uji	Use Case	Metode Pengujian	Kriteria Evaluasi Hasil Pengujian
UC-1.	Mengelola Pengajar	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus profil pengajar.
UC-2.	Mengelola Siswa	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus profil siswa.
UC-3.	Mengelola subjek	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus subjek
UC-4.	Mereset Password	Black Box	Sistem dapat mereset password baik password pengajar maupun siswa.
UC-5	Mengelola Bab	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus bab yang ada pada suatu mata kuliah.
UC-6.	Mengelola Tingkat Kesulitan	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus tingkat kesulitan yang ada pada masing-masing bab.
UC-7.	Mengelola Aturan	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus aturan yang akan digunakan untuk proses adaptif.
UC-8.	Mengelola Pertanyaan	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus pertanyaan kuis.
UC-9.	Mengelola Kuis	Black Box	Sistem dapat menambah, mengupdate, menampilkan dan menghapus konfigurasi kuis dari setiap subjek.

UC-10.	Menampil-kan Histori Hasil Kuis siswa	Black Box	Sistem dapat menampilkan histori hasil kuis siswa
UC-11.	Mengambi l Kuis	Black Box	Sistem dapat me-load sebuah kuis dengan pertanyaan, konfigurasi dan aturan yang sudah dibuat oleh pengajar

Berdasarkan hasil pengujian dan hasil yang didapat dari fungsi masing-masing use case dapat disimpulkan bahwa penerapan arsitektur *model-view-controller* (MVC) dalam rancang bangun sistem kuis online ini dinyatakan berhasil.

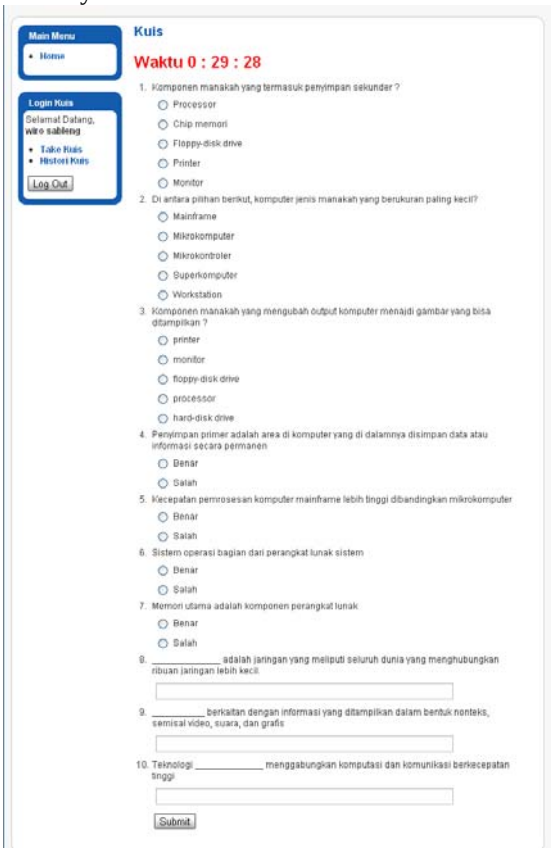
5.2 Pembahasan

Pengembangan sistem kuis online adaptif ini menggunakan arsitektur *model-view-controller* (MVC), sehingga harus mengikuti framework yang diterapkan dalam arsitektur tersebut. Komponen yang dihasilkan berupa komponen pengajar, siswa, semester, mata kuliah, bab, tingkat kesulitan, kuis, soal, aturan (rule), dan histori.

Pada saat memulai kuis, sistem akan melakukan pengecekan, jika siswa merupakan pertama kalinya mengambil kuis untuk mata kuliah yang dipilih, maka akan diarahkan mengikuti *pretest* terlebih dahulu. *Pretest* ini akan menentukan bab dan tingkat kesulitan siswa memulai kuis ini. Hal ini tidak diperlukan siswa untuk selalu memulai dengan tingkat kesulitan level 1. Jika siswa bukan pertama kali mengakses kuis ini , sistem mendapatkan informasi yaitu bab, nilai dan tingkat kesulitan terbaru, seperti ditunjukkan dalam Gambar 6. Berdasarkan informasi yang diberikan, sistem kemudian memilih sekelompok pertanyaan berikutnya yang sesuai dengan tingkat kesulitan yang direkomendasikan dan jumlah pertanyaan yang akan disajikan (dari konfigurasi kuis).

Gambar 6. Halaman siswa ketika mengambil kuis

Proses selanjutnya setelah siswa menjawab dan mengirim jawabannya, seperti yang ditunjukkan dalam Gambar 7, sistem akan melakukan pemeriksaan dan menghitung hasilnya. Sistem kemudian mengupdate dengan informasi terbaru seperti hasil siswa (skor kuis), bab dan tingkat kesulitan terbaru, tanggal kuis diambil dan mata kuliahnya.



Gambar 7. Halaman soal kuis

Langkah selanjutnya, sistem akan mencari tingkat kesulitan siswa berikutnya yang direkomendasikan berdasarkan aturan yang didefinisikan oleh pengajar. Informasi ini akan menentukan apakah siswa harus pergi ke suatu tingkat kesulitan yang lebih tinggi , tetap atau lebih rendah dari tingkat kesulitan saat ini, seperti yang ditunjukkan dalam Gambar 8. Proses ini akan terus berlanjut sampai siswa tersebut memutuskan untuk menghentikan kuis atau siswa telah mencapai tingkat kesulitan tertinggi dari bab terakhir.

Gambar 8. Halaman rekomendasi kuis berikutnya

Sistem kuis online adaptif yang dihasilkan dapat menangani tahapan penilaian secara adaptif serta mampu memberikan umpan balik kepada peserta didik (siswa) setelah mengikuti penilaian tersebut. Hal ini berdasarkan dari hasil implementasi dan

pengujian yang dilakukan terhadap sistem ini. Umpan balik yang didapatkan siswa setelah mengikuti kuis ini, adalah siswa dapat mengetahui kekuatan dan kelemahan yang mereka miliki melalui hasil tiap kuis yang mereka kerjakan, ditambah tingkat pengetahuan yang mereka miliki saat itu. Siswa akan semakin mempunyai motivasi untuk belajar lebih giat, sehingga dapat meningkatkan tingkat pengetahuan mereka.

6. KESIMPULAN

Penerapan arsitektur *model-view-controller* (MVC) dalam rancang bangun sistem kuis online adaptif ini dapat meningkatkan modularitas dan reusabilitas dari sistem. Hal ini dimungkinkan karena *source code* menjadi lebih rapi dan pemisahan antara logika bisnis dan antarmuka pengguna yang lebih eksplisit. Kesimpulannya dengan menggunakan arsitektur ini, kompleksitas dari kode dalam perangkat lunak dapat dikurangi secara signifikan, dengan demikian, meningkatkan fleksibilitas dan modularitas sistem perangkat lunak.

Penerapan teknik pertanyaan adaptif menghasilkan urutan dinamis pertanyaan yang dihasilkan tergantung pada respon peserta didik, dengan kata lain, jawaban peserta didik menentukan seri pertanyaan berikutnya. Proses untuk menentukan pertanyaan-pertanyaan berikutnya, adalah dipicu oleh pengaturan yang dilakukan pengajar.

PUSTAKA

Alotaiby, F.T.; & Chen, J. X. 2005. *Generic Summative Assessment Functional Model*. IEEE.

Balani, Naveen. 2002. *Web services architecture using MVC style*, (Online), (http://www.webifysolutions.com?subject=Web services architecture using MVC style, diakses tanggal 1 April 2010)

Burbeck, Steven. 1992. *Application Programmings in Smaltalk's 80 ™: How To Use MVC*, (Online), (http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html, diakses tanggal 10 Maret 2010).

Cheng, Q. & Kinshuk. 2004. *Application of Adaptivity in Quiz Systems*, (Online), (http://www.col.org/pcf3/Papers/PDFs/Cheng_Kinshuk.pdf, diakses tanggal 26 April 2010).

Deacon, John. 2009. *Model-View-Controller Architecture*, (Online), (http://www.jdl.co.uk/briefings/index.html/#mvc, diakses tanggal 10 Maret 2010)

Gulzar, Nadir. 2002. *Fast track to struts: what it does and how*, (Online), (http://media.techtarget.com/tss/static/articles/content/StrutsFastTrack/StrutsFastTrack.pdf, diakses tanggal 1 April 2010)

Guouli, et al. [Gouli, E., Papanikolaou, K., Grigoriadou, M]. 2002. *Personalizing Assessment in Adaptive Educational Hypermedia Systems*, (Online), (http://hermes.di.uoa.gr/lab/CVs/papers%5Cpapanikolaou%5Cgpg_AH2002.pdf, diakses tanggal 27 April 2010).

Kules, B. 2000. *User Modeling for Adaptive and Adaptable Software Systems*, (Online), (http://www.otal.umd.edu/UUGuide/wmk/. diakses tanggal 25 April 2010).

QuestionMark; & League. 2004. *An Assessment Framework for the Community College*, (Online), (http://www.league.org/publication/whitepapers/files/0804.pdf, diakses tanggal 27 April 2010).

Quinn, D.; & Reid, I. 2003.*Using Innovative Online Quizzes to Assist Learning*, (Online), (http://ausweb.scu.edu.au/aw03/papers/quinn/paper.html, diakses tanggal 27 April 2010).

Santally, M. I.; & Senteni, A. 2005. *Adaptation Models for Personalization in Web-based Learning Environments*, (Online), (http://72.14.235.104/search?q=cache:0lBXmKxULEwJ:pppjj.usm.my/mojit/articles/pdf/April05/01-Santally-revised-typeset.pdf+adaptation+santally+models&hl=en&ct=clnk&cd=1&gl=my, diakses tanggal 12 Mei 2010).

Satish. 2004. *Model View Controller (MVC) Architecture*, (Online), (http://www.dotnetspider.com/resources/316-Model-View-Controller-MVC- architecture.aspx, diakses tanggal 13 April 2010).

Weibelzahl, S. 2002. *Evaluation of Adaptive Systems*, (Online), (http://www.idemployee.id.tue.nl/g.w.m.rauterberg/amme/weibelzahl(2002).pdf, diakses tanggal 24 April 2010).