ISSN: 1978 - 8282

The Concept of Model-View-Controler (MVC) as Solution Software Development

Ermatita [1]
Huda Ubaya [2]
Dwirosa Indah [3]

ermatitaz@yahoo.com, hudaubaya@yahoo.com, dwirosa@gmail.com

Diterima: 9 Maret/Disetujui: 19 Maret 2010

Abstract

Software development is a complex task and requires adaptation to accommodate the needs of the user. To make it easier to changes the software, in maintenance, now has developed concept in the development of the software, the model-view-controller pattern, which is the architecture that can help facilitate in the development and maintenance of software, because in this architecture for a three-layer model, namely, the view and controller in development done independently, so that it can provide convenience in the development and maintenance. In addition, this architecture can also view a simple and interesting for the user. Software system on-line test is software that requires interaction with the user, and maintenance of adaptive software. Because the test system on-line requires the development of software to accommodate the needs of this growing quickly. This paper to analyse the Model-View-Controller and try development, to apply it in the development of software system test on-line.

Keywords: *Model-view controller, architecture, pattern , on-line test*

ABSTRAK

Pengembangan perangkat lunak adalah tugas kompleks dan membutuhkan adaptasi untuk mengakomodasi kebutuhan pengguna. Untuk membuat konsep dan perubahan perangkat lunak, dalam pemeliharaan, sekarang telah dikembangkan lebih mudah dalam pengembangan perangkat lunak, pola model-view-controller, yang merupakan arsitektur yang dapat membantu memfasilitasi dalam pengembangan dan pemeliharaan perangkat luna. Hal ini, karena dalam arsitektur model tiga-lapis, yaitu: tampilan dan pengontrolan dalam pembangunan dilakukan secara independen, sehingga dapat memberikan kemudahan dalam pengembangan dan pemeliharaan. Selain itu, arsitektur ini juga dapat melihat hal-hal yang sederhana dan menarik bagi pengguna. Software sistem on-line test

1. **Dosen Fakultas Ilmu Komputer, Universitas Sriwijaya Palembang**
   Jl. Raya Palembang - UNSRI Km. 32 Ogan Ilir
2. **Dosen Fakultas Ilmu Komputer, Universitas Sriwijaya Palembang**
   Jl. Raya Palembang - UNSRI Km. 32 Ogan Ilir
3. **Dosen Fakultas Ilmu Komputer, Universitas Sriwijaya Palembang**
   Jl. Raya Palembang - UNSRI Km. 32 Ogan Ilir

adalah perangkat lunak yang memerlukan interaksi dengan pengguna, dan pemeliharaan perangkat adapt.f. Karena sistem ujian on-line memerlukan pengembangan perangkat lunak untuk mengakomodasi kebutuhan ini berkembang dengan cepat. Makalah ini untuk menganalisis Model-View-Controller dan mencoba pembangunan, untuk menerapkannya dalam pengembangan perangkat lunak sistem pengujian on-line.

***Kata Kunci**: Model-view controller, arsitektur, pola, on-line test*

INTRODUCTION

Software development is a complex task. Software development process, from concept to implementation, known by the term System Development Life Cycle (SDLC) which includes stages such as requirement analysis, design, code generation, implementation, testing and maintenance (Pressman, 2002; 37-38). In the software development also requires architecture or pattern that can help in the development of software.

Currently, many software development utilizing a concept of programming by using the Model-view-controller pattern, which is the architecture with a lot of help in the development of software that is easy to maintenance, especially those based interaction with the GUI (Graphical User Interface) and the web. This is in accordance with the statement Ballangan (2007) : "MVC pattern is one of the common architecture especially in the development of rich user interactions GUI Application. Its main idea is to decouple the model. The interaction between the view and the model is managed by the controller. "

In addition Boedy, B (2008) [1] states that: MVC is a programming concept that applied to many of late. By applying the MVC to build an application will be impact to ease at the time the application enters the maintenance phase. Development process and integration is becoming easier to do. Basic idea of MVC is actually very simple, namely to try to separate model, view, and controller.

The concept of software development with the architectural model, controller and view this is very helpful in the development of software test on-line. This is because software development with the concept of model view controller, and this can help make it easier to do maintenance and make the look interesting, so that user interaction with the software more attractive and easier.

This study focused on the concept of software-based GUI to apply the Model-View-Controller pattern is applied to the development of software Testing On-line at the Computer Science Faculty of Sriwijaya University considerations related to the concept in terms of quality, flexibility and ease of maintenance (expansion or improvement), as cited by many of the literature.

From the background that has been described above will be conducted the research with the title: "The concept of Model-View-Controler (MVC) in the case study solutions Software development test on-line"

PROBLEM FORMULATION

Research was conducted in order to solve the problem how architecture Model-View-Controller (MVC) in the ease of making the software, in this case study application to take the exam online.

OBJECTIVES

This study aims to understand the development of Software architecture model with the pattern model-view-controller with on a case study to implement the software system on-line test.

DEFINITIONS

Popadyn, 2008 [8] defines: Model - View - Controller is an Architectural pattern used in software engineering. Nowadays such complex patterns are Gaining more and more popularity. The reason is simple: the user interfaces are becoming more and more complex and a good way of fighting against this complexity is using Architectural patterns, such as MVC.

MVC pattern consists of three layers as in research by Passetti (2006): The pattern separates Responsibilities across the three components, each one has only one responsibility.

a.The view is responsible only for rendering the UI elements. It gives you a presentation of the model. In most implementations the view usually gets the state and the data it needs to display directly from the model.

b.The controller is responsible for interacting between the view and the model. It takes the user input and figures out what it means to the model.

c.The model is responsible for business behaviors, application logic and state management. It provides an interface to manipulate and retrieve its state and it can send notifications of state changes to observers. Each layer has the responsibility of each of each integrated with one another. The MVC abstraction can be graphically reprensented as follows.

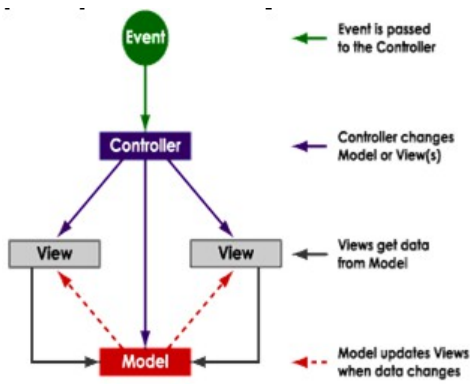

Figure 1. MVC abstraction

Events typically cause a controller to change a model, or view, or both. Whenever a controller changes a model's data or properties, all dependent views are automatically updated. Similarly, whenever a controller changes a view, for example, by revealing areas that were previously hidden, the view gets data from the underlying model to refresh itself. Otherwise, the common workflow of the pattern is shown on the next diagram.

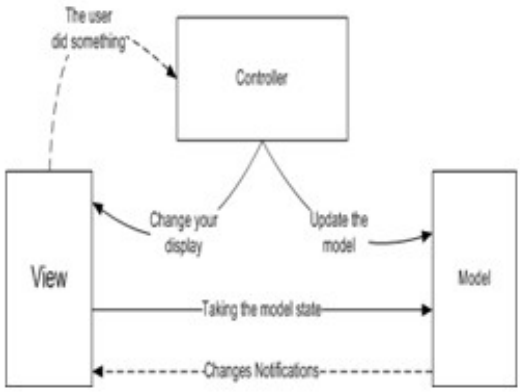

Figure 2. Common workflow MVC pattern

Control flow generally works as follows:

a. The user interacts with the user interface in some way (e.g. presses buttons, enters some input information, etc.).

b. A controller handles the input event from the user interface.

c. The controller notifies the model of the user action, possibly resulting in a change in the model state

d. A view uses the model to generate an appropriate user interface. The view gets its own data from the model. The model has no direct knowledge of the view.

e. The user interface waits for further user interactions, which then start a new cycle.

MODEL VIEWE CONTROLLER ARCHITECTURE

On the development of software architecture with the pattern or model-view-controller, the software development divide into three parts, namely the model, view and controller.

Biyan (2002) points out: MVC is a design pattern that enforces the Separation between the input, processing, and output of an application. To this end, an application is divided into three core components: the model, the view, and the controller. Each of these components handles a discreet set of tasks.

a. Model

Model here as a representation of the data involved in a transaction process. Each time the method / function of an application need to access to data in a, the function / method is not directly interaction with the source data through the model but should be first. In this model only allowed to interact directly with the source data.

b. View

View as the presentation layer or user interface (display) for the user of an application. Data needed by the user will be formatted in such a way that can be run and presented with the view that the format is adjusted to the user requirement.

c. Controller

Controller is logic aspect of an application. Controller will determine the process bussiness of applications are built. Controller will respond to each user's input with the conduct of the model and view so that the appropriate request / demand from the user can be met well.

Each layer are interconnected and mutual dependence of each other, this is like that disclosed by Anderson, DJ (2000), As we are building each View on demand, it must request data from the model every time it is instantiated. Hence,

there is no notification Model to View. The creation of the View object which must demand any necessary data from the model.

Views and Controllers together can be considered the Presentation Layer in a Web Application. However, as we will see it is easy to cleanly separate Views from Controllers in a Server-side Web Application.

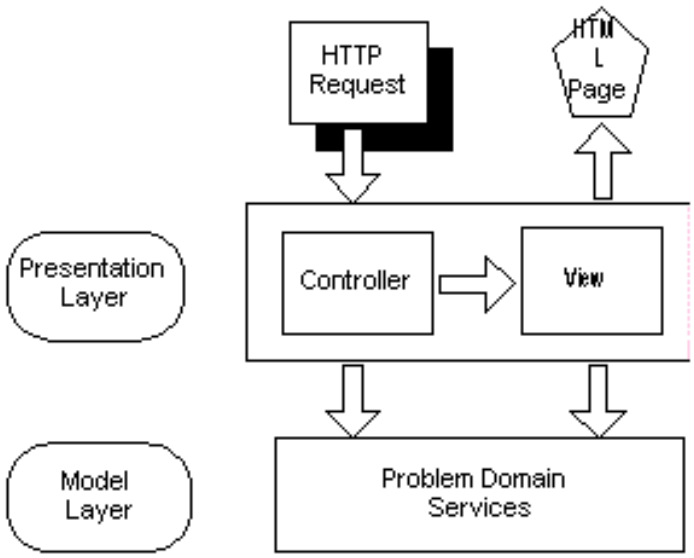

Figure 3. Client to Presentation Layer Interaction showing the MVC Separation at the Server

The Model, on the other hand, is separate from the Presentation Layer Views and Controllers. It is there to provide Problem Domain services to the Presentation Layer including access to Persistent Storage. Both View and Controller may message down to the model and it in turn will send back appropriate responses.

Some of the MVC in field research has been done, among others:
Sauer in the research entitled "MVC-Based Modeling Support for Embedded Real-Time Systems" states that "Effects of real-time requirements on the model, view, controller and communication components have to be identified, eg in the scenario for signalling to a warning message and handling it by the user or in the case of exception handling. It has to be answered, which time constraints apply to each component "effects in real-time requirement on the model view controller can be identified so that it can be signed quickly.

Ogata in their paper describes the design of the framework that the Model-View- Controller is called Event-driven MVC-Active is based on the active object.

On this model treats each input by using the Event-driven mechanism and the process of placing objects in the active display. This model is very suitable for applications that focus on the GUI. This model has been implemented to the Smalltalk.

Naturo, et all, (2004) propose a design pattern that supports the construction of adaptable simulation software via an extension of the Model / View / Control design pattern. The resulting Model / Simulator / View / Control pattern incorporates key concepts from the DEVS modeling and simulation methodology in order to promote a Separation of modeling, simulation, and distributed computing issues. The advantage of this simulation approach to software design is considered in the context of other documented attempts to promote component based simulation development [6].

Veit (2003) propose to use the model-view-controller paradigm as a benchmark for AOSD approaches, since it combines a set of typical problems concerning the Separation and integration of contens. [11]

Morse, SF et all(2004) introduce Model-View-Controller Java Application Framework. This paper propose a simple application framework in Java that follows a Model-View-Controller design and that can be used in introductory and core courses to introduce elements of software engineering . Although it is focused on applications having a graphical interface, it may be modified to support command-line programs. The application framework is presented in the context of development tools Apache Ant and JUnit. [5]

Software development through the MVC approach, maintenance and evolution of the system more easy to do, with the divide in layers. The layers in the software divide into three layer.

By applying the concept of MVC, the source code of an application to be more tidy and easier to maintenance and developed.

In addition, the program code has been written that can be used again for other applications by changing only some. Basically, the MVC to separate the data processor referred to as a model and user interface (UI) or HTML template in this case called the View. Mean while, as the fastener Controller Model "View to handle the request so that the user data is displayed correctly.

Online Testing

Testing activities as one of the activities in the teaching-learning can be done anywhere with the help of information technology. This activity is called with the test on-line.

In the test conducted on-line, the need for the many variations of problems and supply problems are very important addition to the maintenance of the software is needed to accommodate the needs of adaptation to the development of the user. Effectiveness and level of security required in both documents the shoptalk addition, the tests also required a good appearance and reliable in showing shoptalk test. Is

required for a concept development and architecture in both software development support on-line test this.

RESEARCH METHODE

Steps will be done in this research are: study of literature on the concept of architectural pattern Model-View-Controller, and implementing MVC architectural pattern in development software test on-line.

RESULTS AND DISCUSSION

1. Analysis of Needs

The results of the analysis needs in the form of Functional Requirement have 3 main menus: the login process, the main page, the lecturers and the student Menu

2. MVC Architecture Concept Analysis Software Testing Online

Architecture Model-View-Controller pattern is that can help build the project more effectively. In the development of this pattern is done with divide component of Model, View and Controller in the development of software. Divide is useful to separate the parts in the application so that the ease in application development and maintenance. This is in line with the expression: Model-View-Controller design software to assist with the needs divide similar. Model View Controller is a class analysis to the stages of the design phase. [7] On a system equipped with the software, the changes are often the user interface. User interface is the part that dealt directly with the user and how it interacts with the application, the focus point made to conversion based on ease of use.

Business-logic in a complex user-interface to make conversion to the user interface became more complex and simple mistakes occur. Changes one part has the potential of the overall software. MVC can provide a solution to the problem by dividing the pattern into sections separate, Model, View and Controller, split between the part and create a system of interaction between them

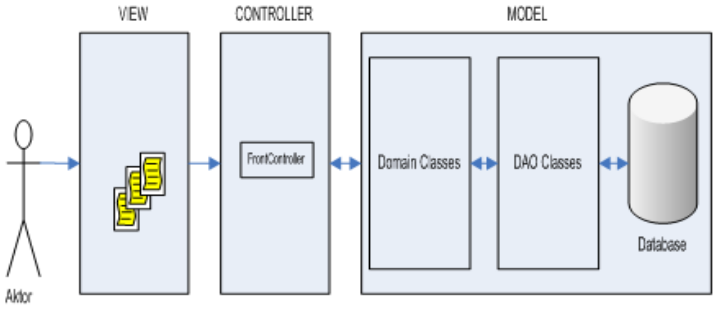

Figure 5. MVC architecture is applied to the software test online

3. Model layer

Layer model in the MVC pattern that represents the data used by applications as business processes associated it. Domain Model is a representation of the model layer. Whole class at the model are in a layer model, including the class that supports it (DAO class).

4. View layer

This layer contains the entire details of the implementation of the user interface. Here, the components provide a graphical representation internal application process flow and lead to the application user interaction. There is no other layer that interacts with the user, only the View.

On the software Online Testing System which is being developed, a layer view of the files containing the display for the user that consists mainly of HTML script.

5. Controller layer

Controller layer in MVC provides detailed program flow of each layer and transition, will be responsible for gathering events made by the user from the View and update components of the model using data entered by the user.

On the software Online Testing System which is being developed, there is only one controller class that is Front Controller class. Use one Controller class is because the scope of the software that is still small, so the controller does not need separation.

CONCLUSION

Results from this research are: development of software can be divide in several layers, divide layer in the development of software architecture with the concept of Model-View-Controller can assist in system maintenance and evolution, and software development Testing system architecture apply online with the Model-Controller-View very helpful in further development, because the software system test online this very need for reliability Users interface.

REFERENCES

[1] Boedy, B, 2008, Model-View-controller, avalable on http; / / MVC / Model-view-controller.html

[2] David J. Anderson, 2000, Using MVC Pattern in Web Interactions, http:// www.uidesign.net/Articles/Papers/UsingMVCPatterninWebInter.html

[3] Hariyanto, B, 2004., "Object-oriented Systems Engineering", Bandung: Information Cackle, B, 2002, the MVC design pattern brings about better organization and code reuse, Oct 30, 2002 8:00:00 AM

[4] Mathiassen, L, Madsen, Andreas. M, Nielsen, Peter. A, and the Stage, A, 2000, "Object Oriented Analysis & Desig. Issue 1. Forlaget Marko, Denmark.

[5] Morse, SF, and Anderson, CL, 2004, Design and Application Introducing Software Engineering Principles in introductory CS Courses: Model-View-Controller Java Application Framework http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1. 1.73.9588

[6] Nutaro, A and Hammonds, P, 2004, combining the Model / View / Control Design Pattern with the DEVS Formalism to achieve Rigor and Reusability in Distributed Simulation, JDMS, Vol. 1, Issue 1, April 2004 Page 19-28, © 2004 The Society for Modeling and Simulation International

[7] Nugroho. A, 2002, "Analysis and Design Object-oriented System," Bandung: Information. Passeti, 2006, The MVC Pattern © 2006, P & P Software www.pnp software.comhttp: / / images.google.co.id / imgres? Imgurl = http:/ /www.pnp-software.com/eodisp/images/mvc -original.

[8] Popadiyn, P, "Exploring the Model - View - Controller (MVC) pattern," Pencho Popadiyn posted by on Dec 17, 2008

[9] Pressman, RS, 2002, "Software Engineering: Practitioners Approach" Translated by: CN. Harnaningrum (Book 1). Yogyakarta: ANDI. Sutabri, T, 2004, "Analysis of Information Systems", Yogyakarta: ANDI.

[10] O'Brien, James A. (2003). Introduction to information systems: Essentials for the e-business enterprise, edition to-11. McGraw-Hill, Boston.

[11] Veit, and M Herrmann, S, 2003, "Model View Controller and Object Teams: A Perfect Match of Paradigms", avalaibale on http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.5963

[12] Model-View-Controller Pattern, Copyright © 2002 eNode, Inc.. All Rights Reserved.http: / / www.enode.com / x / markup / tutorial / mvc.html

[13] MVC Pattern (MVC Framework) http://209.85.173.132/search?q=cache:e2SN1LcxrZEJ:www.javapassion.com/j2ee/MVCPatternAndFrameworks_speakernoted.pdf + MVC + pattern & hl = en & ct = clnk & cd = 14 & gl = id

[14] ——, 2006, MVC Pattern (MVC Framework) http://www.javapassion.com/j2ee/MVCPatternAndFrameworks_speakernoted.pdf

[15] http://www.mercubuana.ac.id/sistem.php, access in 4 september 2008 http://id.wikipedia.org/wiki/Perangkat_lunak, "software", tgl.4 access in September 2007.

[16] file: / / / D: /% MVC 20teori/Mengenal% 20MVC% 20di% 20Zend% 20Framework% 20% 97% 20Pemrograman% 20Web.htm

ISSN: 1978 - 8282

PEDOMAN PENULISAN

Lingkup Jurnal. Tulisan yang dapat dimuat adalah yang mengkaji masalah yang berhubungan dengan bidang ilmu komputer dan teknologi informasi, baik ilmudasar maupun aplikasinya.

Jenis tulisan. Tulisan dapat berupa laporan/ hasil penelitian atau makalah ilmiah bukan penelitian seperti laporan studi kasus atau kajian pustaka komprehensif. Tulisan ilmiah/ penelitian dapat merupakan hasil a) Pengembangan, b) Penemuan, dan c) Pembuktian

A. Laporan penelitian minimal harus memuat bagian abstrak, pendahuluan (latar belakang, t ujuan, hipotesis, konsep2 utama), metodologi, hasil dan pembahasan, kesimpulan dan pustaka.

B. Makalah ilmiah bukan penelitian minimal harus memuat bagian abstrak, pendahuluan, pembahasan, kesimpulan dan pustaka.

Nama Penulis. Ditulis tanpa gelar dan jabatan, disebutkan nama Lembaga dan alamatnya serta alamat e-mail.

Bahasa. Ditulis dalam bahasa Indonesia atau bahasa Inggris dengan memperhatikan kaidah-kaidah bahasa ragam ilmiah. Khusus untuk yang menggunakan bahasa Indonesia, hindari penggunaan kata ganti orang.

Panjang Tulisan. Panjang tulisan 10-15 halaman A4 spasi single termasuk tabel dan gambar serta lampiran, dengan jenis huruf Times New Roman, font 11.

Abstrak. Panjang abstrak maksimum 250 kata, dalam satu paragraf, dilengkapi dengan kata-kata kunci pada bagian akhir abstrak. Abstrak memuat latarbelakang, metodologi, hasil dan kesimpulan. Abstrak tidak bersifat matematis, tidak berisi saran dan harapan, tidak ada kutipan. Kata kunci (*keywords*) adalah kata-kata penting yang digunakan untuk mengidentifikasikan isi dokumen, dapat berupa metode/alat yang dipakai, variabel yang diteliti atau substansi penelitian. Abstrak ditulis dalam bahasa Indonesia dan Inggris.

Tabel dan gambar. Tabel dan gambar harus diberi nomer dan judul lengkap serta harus diacu dalam tulisan. Gambar dan tabel dalam format hitam putih.

Persamaan. Persamaan matematik harus diberi nomer urut dalam kurung biasa ,(x), dengan penulisan rata kanan.

Setiap makalah diwajibkan untuk mengutip sumber pustaka yang berasal dari jurnal ilmiah nasional maupun internasional.

Kutipan. Setiap kutipan harus menyertakan sumbernya yang ditulis pada

kutipan, yaitu dengan menuliskan nama belakang pengarang pertama (jika pengarang lebih dari satu: dituliskan nama pengarang pertama et al), dan tahun terbit. Contoh : .....menurut Angel (2003), atau .....(Angel, 2003), atau ..... Chen et al(2007)

Pustaka. Pustaka disusun terurut berdasarkan nama belakang pengarang dan hanya memuat pustaka yang dikutip dalam tulisan. Nama pengarang ditulis tanpa gelar, jika ada nama tengah dan belakang, disingkat. Contoh :

Buku.

1. Angel (2003). *Interactive Computer Graphics: A Top-Down Approach Using CpenGL.* Third Edition. London: Pearson Education.
2. Irianto (2004). *Embedding Pesan Rahasia Dalam Gambar*. Bandung: Institut Teknologi Bandung.

Tulisan/artikel dalam buku.

1. Bolton, MA: Anker Publisher Inc, 144-158. Yudhana, A (2007). Desain Routing Trafik Jaringan Telekomunikasi dengan algoritma Genetik, dalam Wibowo, T.A (Ed), *Berbagai Makalah Sisitem Iɴformasi,*
2. Olanivan, B.A (2004). Computer-mediated Communication as an instruction learning tool: course evaluation with communication students, in Comeaux, P(Eds), *Assessing Online Teching&Learning*,
3. *Prosiding Koɴferensi Nasional Sistem Iɴformasi 2007*, Bandung : Departemen Teknik Informatika, Sekolah Tinggi Teknologi Telkom, 233-238.

Jurnal.

1. Goyal, D.P (2007). Information Systems Planning Practices in Indian Public Enterprises. *Iɴformation Management & Computer Security,* 15(3), 201-213

Sumber online.

1. Chen, CC., Wu, J., Yang, SC. (2007). *The Eɸficacy cf online cooperative learning systems. The perspective of task-technology fit.* Diakses pada 20 Mei 2007 dari : http: //www.emeraldinsight .com/1065-0741.htm.
2. Marques, O.,Baillargeon, P (2007). *Design cf multimedia traɸfic classifier for snort.* Diakses pada 2 Juni 2007 dari : http: // www.emeraldinsight.com/ 0968-5527.htm.

Sekretaris Redaksi

Henderi, M. Kom.

FORMULIR PERSETUJUAN PEMBUATAN ARTIKEL JURNAL	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-003
PENANGGUNG JAWAB	**MENYETUJUI**
KETUA STMIK RAHARJA	
DIREKTUR AMIK RAHARJA INFORMATIKA	
TENTANG/PERIHAL/JUDUL	
Judul terlampir :	
Abstraksi terlampir :	
BAGIAN PENULIS	**MEMOHON**
Nama Penulis Naskah/Pengarang 1	
Nama Penulis Naskah/Pengarang 2	
Nama Penulis Naskah/Pengarang 3	
Nama Editor/Penyunting	
Nama Penyunting/Picture Layout & Artistik	
KETUA EDITOR	**MEREKOMENDASIKAN**
Reviewer 1	
Reviewer 2	

FORMULIR KRITERIA DAN BOBOT PENILAIAN KARYA TULIS ILMIAH	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-001

Kode Judul : _____

Judul Karya Tulis Ilmiah : _____

_____

_____

Reviewer : ☐ Mitra Bestari ☐ Dewan Redaksi

Nama Reviewer : _____

NO	KRITERIA (NILAI MAKSIMAL)	INDIKATOR PENILAIAN	HASIL PENILAIAN NARATIF DAN SARAN	NILAI SETIAP KRITERIA
1.	**JUDUL (5)**	A. Maksimal 14 (empat belas) kata dalam Bahasa Indonesia atau 10 (sepuluh) kata dalam Bahasa Inggris **(1)** B. Relevan dengan isi sangat jelas **(2)** C. Relevansinya dengan permasalahan sangat jelas **(2)**		
2.	**ABSTRAK (5)**	A. Dalam Bahasa Indonesia dan Bahasa Inggris yang baik **(5)** jika hanya dalam Bahasa Indonesia yang baik atau Bahasa Inggris yang baik **(2.5)** B. Format sesuai dengan pedoman **(1)** C. Isi : Latar belakang metode, hasil, dan kesimpulan tertuang dengan kalimat yang jelas **(4)**		

3.	**SISTEMATIKA (15)**	A. Sesuai dengan Pedoman **(5)** B. Ada Instrumen pendukung (gambar, grafik) dan sangat relevan **(5)** C. Daftar pustaka : dominan terbitan 10 (sepuluh) tahun terakhir dan pustaka primer **(5)**		
4.	**SUBSTANSI (70)**	A. Data/informasi telah diolah dengan sangat baik **(10)** B. Relevansi latar belakang dan pembahasan sangat jelas **(15)** C. Analisis dan sintesis atau pembahasan sangat baik **(25)** D. Kesimpulan : sangat jelas relevansinya dengan latar belakang dan pembahasan, dirumuskan dengan singkat **(20)**		
TOTAL NILAI MAKSIMAL				

Hasil Penilaian:*

☐ Diterima ☐ Diterima dengan baik ☐ Ditolak

Keterangan:
**** Hasil penilaian: nilai total > 75, makalah diterima***

Catatan untuk redaksi pelaksana:
1. Tulisan yang dikirim kepada pemeriksa, nama penulisnya ditutup, dan diganti nomer kode.
2. Setiap tulisan diperiksa oleh dua orang, satu orang dari dewan redaksi dan satu orang dari mitra bestari.

ISSN: 1978 - 8282

FORMULIR EDITOR BAHASA KARYA TULIS ILMIAH	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-004

Kode Judul : _____

Judul Karya Tulis Ilmiah : _____

_____

_____

Editor : ☐ Editor Bahasa

Nama Editor : _____

Saya yang bertanda tangan di bawah ini menyatakan semua tulisan/kalimat sesuai dengan kaedah EYD, atau sesuai dengan kaedah-kaedah tata bahasa.

Catatan:

Rekomendasi: *

☐ Diterima ☐ Diterima dengan perbaikan ☐ Ditolak

Tangerang, _____

_____

Editor

Keterangan:

**** Rekomendasi diisi berdasarkan hasil pemeriksaan editor***

ISSN: 1978 - 8282

FORMULIR EDITOR LAYOUT DAN ARTISTIK KARYA TULIS ILMIAH	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-005

Kode Judul : _____

Judul Karya Tulis Ilmiah : _____

_____

_____

Editor : ☐ Editor Layout dan Artistik

Nama Editor : _____

Saya yang bertanda tangan di bawah ini menyatakan bahwa desain untuk layout dan artistik sudah format jurnal CCIT yang ditentukan.

Catatan:

Rekomendasi: *

☐ Diterima ☐ Diterima dengan perbaikan ☐ Ditolak

Tangerang, _____

_____

Editor

Keterangan:

**** Rekomendasi diisi berdasarkan hasil pemeriksaan editor***

ISSN: 1978 - 8282

FORMULIR PENYELESAIAN ARTIKEL	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-006

TENTANG/PERIHAL/JUDULARTIKEL:

BAGIAN	KETERANGAN
Nama Penulis Naskah/Pengarang 1	☐ Lengkap
Nama Penulis Naskah/Pengarang 2	☐ Lengkap
Nama Penulis Naskah/Pengarang 3	☐ Lengkap
Nama Editor/Penyunting	☐ Lengkap
NamaPenyunting Picture/Layout & Artistik	☐ Lengkap
Nama Peninjau (Reviewer) 1	☐ Lengkap
Nama Peninjau (Reviewer) 2	☐ Lengkap
Nama Percetakan	☐ Lengkap

Formulir ini menyatakan bahwa artikel ini telah dinyatakan layak untuk diterbitkan pada Journal CCIT. Penerbitan adalah sepenuhnya wewenang redaksi mengingat banyaknya artikel yang masuk.

MENYETUJUI

KETUA DEWAN EDITOR

(.....)

SEKRETARIS REDAKSI

(.....)

ISSN: 1978 - 8282

FORMULIR KESEDIAAN MITRA BESTARI JURNAL ILMIAH	Tanggal Revisi : 12 Desember 2007 Tanggal Berlaku : 13 Desember 2007 Kode Dokumen : FM-RHJ-016-002

Yang bertanda tangan di bawah ini :

Nama Lengkap : _____

Jenjang Pendidikan : _____

Bidang Keahlian : _____

Jabatan Fungsional : _____

Pengalaman ReviewerJurnal : Ya / Tidak *), jika Ya sebutkan dimana, kapan nama jurnal yang di review:

_____

Bersedia menjadi reviewer ahli / Mitra Bestari Jurnal Ilmiah yang berada di bawah naungan Perguruan Tinggi Raharja.

Demikian formulir ini saya tanda tangani untuk dapat dipergnakan sebagai mana mestinya.

Tangerang, _____

Mengetahui, Mitra Bestari,

(.....) (.....)

*) Coret yang tidak perlu

Index Judul

Index Penulis

Index Subjek