

***REENGINEERING* APLIKASI MOFLUS
DENGAN PROSES MODEL HORSESHOE**



SKRIPSI

**Disusun Sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Komputer
pada Departemen Ilmu Komputer/ Informatika**

**Disusun oleh:
Yutta Nandiya Putri
24010315130098**

**DEPARTEMEN ILMU KOMPUTER/INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
2019**

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini :

Nama : Yutta Nandiya Putri

NIM : 24010315130098

Judul : *Reengineering* Aplikasi Moflus Dengan Proses Model Horseshoe

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang penuh diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan di dalam daftar pustaka.

Semarang, 18 Oktober 2019



Yutta Nandiya Putri

24010315130098

HALAMAN PENGESAHAN

Judul : *Reengineering* Aplikasi Moflus Dengan Proses Model Horseshoe
Nama : Yutta Nandiya Putri
NIM : 24010315130098

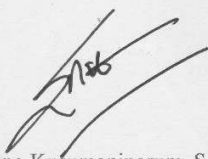
Telah diujikan pada sidang skripsi dan dinyatakan lulus pada tanggal **1 Oktober 2019**.

Semarang, 18 Oktober 2019

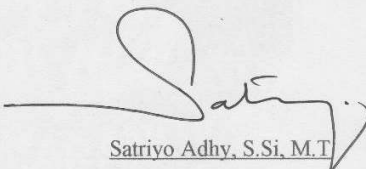
Mengetahui,

Ketua Departemen Ilmu Komputer/Informatika
FSM UNDIP

Panitia Penguji Skripsi
Ketua,


Dr. Retno Kusumaningrum, S.Si, M.Kom

NIP. 198104202005012001


Satriyo Adhy, S.Si, M.T

NIP. 198302032006041002

HALAMAN PENGESAHAN

Judul : *Reengineering* Aplikasi Moflus Dengan Proses Model Horseshoe

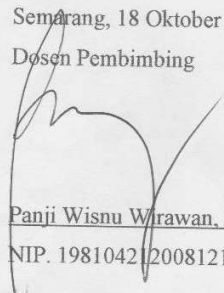
Nama : Yutta Nandiya Putri

NIM : 24010315130098

Telah diujikan pada sidang skripsi pada tanggal **1 Oktober 2019**

Semarang, 18 Oktober 2019

Dosen Pembimbing


Panji Wisnu Wirawan, M.T.

NIP. 198104212008121002

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul “*Reengineering Aplikasi Moflus Dengan Proses Model Horseshoe*”.

Dalam penyusunan laporan skripsi ini penulis mendapat banyak bantuan, bimbingan dan dukungan dari berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Dr. Widowati, S.Si., M.Si., selaku Dekan Fakultas Sains dan Matematika Universitas Diponegoro.
2. Dr. Retno Kusumaningrum, S.Si., M.Kom., selaku Ketua Departemen Ilmu Komputer/ Informatika.
3. Panji Wisnu Wirawan, M.T., selaku Koordinator Skripsi dan Dosen Pembimbing Skripsi.
4. Seluruh pihak yang telah membantu hingga selesainya skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam laporan ini masih terdapat banyak kekurangan, baik dalam penyampaian materi maupun isi dari materi tersebut. Hal ini disebabkan oleh keterbatasan kemampuan dan pengetahuan dari penulis. Oleh karena itu, kritik dan saran yang bersifat membangun sangat penulis harapkan. Semoga skripsi ini dapat bermanfaat bagi penulis dan juga pembaca pada umumnya.

Semarang, 18 Oktober 2019

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Diponegoro, saya yang bertanda tangan di bawah ini:

Nama : Yutta Nandiya Putri
NIM : 24010315130098
Program Studi : Informatika
Departemen : Ilmu Komputer/Informatika
Fakultas : Sains dan Matematika
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan **Hak Bebas Royalti Non-eksklusif (Non-exclusive Royalty Free Right)** kepada Universitas Diponegoro atas karya ilmiah saya yang berjudul:

Reengineering Aplikasi Moflus Dengan Proses Model Horseshoe

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-eksklusif ini Universitas Diponegoro berhak menyimpan, mengalihmedia/ formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/ pencipta dan sebagai pemilik Hak Cipta. Demikian pernyataan ini saya buat dengan sebenarnya.

Semarang, 18 Oktober 2019

Yang menyatakan

The image shows a 6000 Rupiah revenue stamp from the Indonesian government. The stamp includes the text 'METERAI TEMPEL', 'ETN:BCAHF015512661', and '6000 ENAM RIBU RUPIAH'. A handwritten signature is written over the stamp.

Yutta Nandiya Putri

24010315130098

ABSTRAK

MoFluS merupakan sebuah perangkat lunak *smartphone* yang berfungsi sebagai POC (*point-of-care*). Namun terdapat masalah pada MoFluS yaitu sulitnya pengembangan pada kode yang sudah ada. Kesulitan pengembangan disebabkan oleh dependensi yang erat pada kode. Untuk mengurangi dependensi yang erat pada kode, perlu dilakukan perubahan pada struktur MoFluS. Perubahan dilakukan dengan mengubah struktur MoFluS yang tidak MVP (*model-view-presenter*) menjadi MVP. *Reengineering* dilakukan menggunakan proses model *horseshoe*. Pada proses model *horseshoe* dilakukan *reengineering* pada tiga level yaitu pada level konseptual, level fungsional, dan pada level *source code*. Hasil *reengineering* menunjukkan adanya kemudahan pengembangan pada aplikasi. Setelah *reengineering*, dilakukan pengujian JHawk untuk membandingkan kualitas perangkat lunak sebelum transformasi dan sesudah transformasi. Berdasarkan pengujian yang telah dilakukan dengan JHawk 6, hasil *reengineering* aplikasi MoFluS menunjukkan adanya peningkatan nilai *maintainability index* sebanyak 28%.

Kata Kunci: MVP, *reengineering*, proses model *horseshoe*

ABSTRACT

MoFluS is a smartphone software that functions as a POC (point-of-care). But there is a problem with MoFluS, which is the difficulty of developing existing code. Development difficulties are caused by tight dependencies on the code. To reduce the tight dependencies on the code, changes need to be made to the MoFluS structure. Changes are made by changing the MoFluS structure from non-MVP (model-view-presenter) to MVP. Reengineering is done using the horseshoe model process. In the horseshoe model process, reengineering is performed at three levels, namely at the conceptual level, the functional level, and at the source code level. Reengineering results indicate the ease of development in the application. After reengineering, JHawk testing is performed to compare the quality of the software before transformation and after transformation. Based on tests that have been done with JHawk 6, the results of reengineering the MoFluS application show an increase in the value of maintainability index by 28%.

Key Word: MVP, *reengineering*, proses model *horseshoe*

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI.....	ii
HALAMAN PENGESAHAN	ii
HALAMAN PENGESAHAN	iv
KATA PENGANTAR.....	v
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS.....	vi
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	ix
DAFTAR GAMBAR.....	xii
DAFTAR TABEL	xiii
BAB I PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	3
1.3. Tujuan dan Manfaat	3
1.4. Ruang Lingkup	3
BAB II TINJAUAN PUSTAKA	4
2.1. Horseshoe Software Reengineering Process Model	4
2.2. Evolusi Perangkat Lunak (<i>Software Evolution</i>)	5
2.3. Model-View-Presenter (MVP)	6
2.4. Diagram <i>Unified Modelling Language</i> (UML)	7
2.5. JHawk	8
2.6. Maintainability Index (MI)	9

2.7. <i>Cyclomatic Complexity</i>	9
2.8. <i>Halstead Effort</i>	10
2.9. <i>Mobile Fluorescence Spectroscopy (MoFluS)</i>	10
BAB III METODOLOGI	12
3.1. <i>Architecture Recovery</i>	13
3.1.1. <i>Source Text Representation</i>	13
3.1.2. <i>Code Structure Representation</i>	13
3.1.3. <i>Functional Level Representation</i>	13
3.1.4. <i>Architectural Representation</i>	13
3.2. <i>Architecture Transformation</i>	13
3.3. <i>Architecture based Development</i>	14
3.4. <i>Pengujian dengan JHawk</i>	15
BAB IV PEMBAHASAN	16
4.1. <i>Architecture Recovery</i>	16
4.1.1. <i>Source Text Representation</i>	16
4.1.2. <i>Code Structure Representation</i>	16
4.1.3. <i>Functional Level Representation</i>	18
4.1.4. <i>Architectural Representation</i>	18
4.2. <i>Architecture Transformation</i>	19
4.3. <i>Architecture-based Development</i>	20
4.4. <i>Pengujian dengan JHawk</i>	24
4.5. <i>Contoh Maintenance pada MVP</i>	29
BAB V KESIMPULAN DAN SARAN	33
5.1. <i>Kesimpulan</i>	33
5.2. <i>Saran</i>	33

DAFTAR PUSTAKA.....	34
LAMPIRAN – LAMPIRAN	36
Lampiran 1. Daftar <i>File</i> Program MoFluS (sebelum transformasi)	37
Lampiran 2. Daftar <i>File</i> Program MoFluS (sesudah transformasi).....	38
Lampiran 3. <i>Source code</i> LoginController.java	40
Lampiran 4 <i>Source code</i> LoginPresenterImpl.java	45
Lampiran 5. <i>Source code</i> UserInteractorImpl.java.....	46
Lampiran 6. <i>Screenshot</i> Hasil Pengujian di JHawk	48
Lampiran 7. Hasil Ekstrasi Source Code Pada Java Decompiler	49

DAFTAR GAMBAR

Gambar 2.1 <i>Horseshoe reengineering process model</i>	4
Gambar 2.2 Hubungan model, view, dan presenter (Ingeno, 2018).....	7
Gambar 2.3 Contoh diagram alir <i>cyclomatic complexity</i>	10
Gambar 3.1 Garis besar penyelesaian masalah	12
Gambar 4.1 Representasi data dan <i>control flow</i> pada MoFluS sebelum transformasi	17
Gambar 4.1 <i>Package diagram</i> dari komponen <i>view</i> sebelum transformasi	17
Gambar 4.3 <i>Component Diagram</i> MoFluS sebelum transformasi	19
Gambar 4.4 <i>Component diagram</i> MoFluS setelah transformasi	21
Gambar 4.5 <i>Package diagram</i> Login	22
Gambar 4.6 <i>Interface</i> LoginContract.....	23
Gambar 4.7 <i>Interface</i> UserInteractor.....	24
Gambar 4.8 Grafik hasil baris kode.....	25
Gambar 4.9 Grafik hasil pengujian <i>cyclomatic complexity</i>	26
Gambar 4.10 Grafik hasil pengujian <i>halstead effort</i>	27
Gambar 4.11 Grafik hasil <i>maintainability index</i> pada <i>package level</i>	28
Gambar 4.12 Grafik hasil <i>maintainability index</i> pada <i>system level</i>	28
Gambar 4.13 Kelas RGBValue	30
Gambar 4.14 Metode getColorValue untuk mendapatkan hasil perhitungan warna.....	31
Gambar 4.15 Metode savePublicSettings untuk menyimpan konfigurasi pada aplikasi.....	31

DAFTAR TABEL

Tabel 2.1 Komponen <i>package diagram</i> (Pressman & Maxim, 2015).....	7
Tabel 2.2 Komponen <i>component diagram</i> (Rosenberg & Stephens, 2008).....	8
Tabel 4.1 Hasil transformasi kelas	21
Tabel 4.2 Hasil pengukuran baris kode	25
Tabel 4.3 Hasil pengujian <i>cyclomatic complexity</i>	26
Tabel 4.4 Hasil pengujian <i>halstead effort</i>	27
Tabel 4.5 Hasil pengujian <i>maintainability index (package level)</i>	27
Tabel 4.6 Hasil pengukuran <i>maintainability index (system level)</i>	28

BAB I

PENDAHULUAN

1.1. Latar Belakang

POC (*point-of-care*) merupakan salah satu perangkat kesehatan yang penting bagi masyarakat. POC merupakan sebuah istilah untuk perangkat yang berfungsi untuk melakukan pengujian kesehatan yang berlangsung di dekat pasien (Vashist et al., 2015). Keberadaan POC menguntungkan masyarakat karena dapat melakukan pengujian secara efektif dan efisien. Pengujian juga semakin mudah dilakukan dengan adanya perangkat lunak untuk *smartphone* sebagai POC (*Point-of-Care*).

Salah satu perangkat lunak *smartphone* sebagai POC adalah MoFluS (Putra, 2017). Aplikasi MoFluS merupakan aplikasi yang berbasis Android yang berfungsi untuk mendeteksi nilai ekspresi warna dari *fluorescence* melalui kamera pada *smartphone*. *Fluorescence* merupakan salah satu media yang digunakan untuk deteksi kanker (Demchenko, 2015). Fitur utama pada *MoFluS* yaitu pengukuran *fluorescence* secara *real-time*.

Terdapat beberapa masalah yang ada pada MoFluS antara lain sulitnya pengembangan pada kode yang sudah ada, teknologi yang dipakai tidak diperbaharui, alokasi memori yang berlebihan, kondisi *force close* saat aplikasi bekerja, dan dokumentasi yang kurang memadai. Pada aspek sulitnya pengembangan, kode yang sudah ada memiliki dependensi yang erat pada kode yang lain (modul) sehingga dapat mengakibatkan pihak yang melakukan *maintenance* kesulitan dalam mengembangkan kode. Beberapa masalah yang ada terjadi akibat tidak mengalami proses *maintenance* (perawatan) yang memadai.

Proses *maintenance* termasuk dalam pembahasan evolusi perangkat lunak dimana perangkat lunak akan menjalani beberapa perubahan. Perubahan ini disebabkan oleh beberapa hal seperti adanya perkembangan teknologi baru, kebutuhan perangkat lunak yang dapat berubah seiring berjalan waktu, dan perbaikan kerusakan pada perangkat lunak. Terdapat empat kategori *maintenance* perangkat lunak yaitu *preventive*, *corrective*, *adaptive*, dan *perfective* (Mens and Demeyer, 2008).

Tujuan utama dari *perfective* adalah untuk mengubah desain perangkat lunak menjadi lebih baik. *Maintenance* dengan kategori *perfective* diperlukan untuk menangani masalah di MoFluS terutama pada sulitnya pengembangan pada kode yang sudah ada. Dengan menangani masalah tersebut, MoFluS dapat digunakan secara optimal untuk pengembangan selanjutnya.

Untuk menangani masalah kesulitan pengembangan pada kode perlu dilakukan perubahan struktur pada MoFluS. Perubahan yang dilakukan pada struktur perangkat lunak adalah definisi dari restrukturisasi (Eloff, 2002). Restruktisasi merupakan salah satu kegiatan yang termasuk dalam kategori *perfective*. Restruktisasi dilakukan untuk mengubah struktur internal perangkat lunak tanpa mengubah perilaku eksternal (Mens and Demeyer, 2008).

Pada kegiatan *perfective* perlu dilakukan pemahaman terhadap arsitektur atau perilaku pada perangkat lunak tersebut. Kegiatan ini disebut dengan *reverse engineering*. *Reverse engineering* didefinisikan sebagai langkah untuk mengetahui komponen yang ada serta memahami hubungan antar komponen dan menyajikan abstraksi perangkat lunak pada tingkat yang lebih tinggi daripada *source code* (Mens and Demeyer, 2008). Kegiatan *reverse engineering* bersifat pasif dan tidak mengubah keadaan dalam perangkat lunak. Kegiatan yang aktif mengubah perangkat lunak dapat disebut sebagai *reengineering* (rekayasa ulang). *Reengineering* dilakukan untuk mengubah struktur perangkat lunak yang ada menjadi lebih mudah untuk dipelihara (*maintainable*).

Proses dari *reengineering* membutuhkan tahap *reverse engineering*. *Reverse engineering* akan menganalisis produk perangkat lunak untuk mengetahui cara kerjanya. Hasil dari *reverse engineering* digunakan pada revisi dokumentasi perangkat lunak serta mengidentifikasi masalah yang ada, sebagai persiapan untuk *reengineering* (Chikofsky & Cross, 1990).

Berdasarkan proses yang ada yaitu *reengineering* dan *reverse engineering*, maka dapat diketahui bahwa untuk melakukan *reengineering* dapat merujuk pada proses model *horseshoe* (tapal kuda) (Mens & Demeyer, 2008). Pada proses model *horseshoe* dilakukan *reengineering* pada tiga level yaitu pada level konseptual, level fungsional, dan pada level *source code*. Dengan adanya proses yang berjalan pada ketiga level tersebut, dapat dilakukan proses *reengineering* secara menyeluruh pada MoFluS.

Dalam melakukan *reengineering* perlu adanya tujuan transformasi. Tujuan transformasi yang digunakan dalam *reengineering* adalah MVP (*model-view-presenter*). Penggunaan MVP sesuai dengan masalah yang ada yaitu sulitnya mengembangkan kode yang sudah ada. Menurut penelitian yang dilakukan oleh Prabowo et al., (2018), penggunaan MVP dapat meningkatkan *maintainability* dan *modularity* terutama pada Android.

1.2. Rumusan Masalah

Berdasarkan permasalahan yang telah dijelaskan pada latar belakang berkaitan dengan mengembangkan kode yang sudah ada karena memiliki dependensi yang erat maka dapat dirumuskan permasalahan yaitu bagaimana cara melakukan *reengineering* dengan proses model horseshoe pada struktur perangkat lunak MoFluS agar mudah dipelihara?

1.3. Tujuan dan Manfaat

Tujuan dilaksanakannya skripsi ini adalah melakukan *reengineering* dengan proses model horseshoe pada struktur perangkat lunak MoFluS agar mudah dipelihara.

Manfaat dilaksanakannya skripsi ini adalah:

1. Mendapatkan pengetahuan dan pengalaman di bidang *maintenance* aplikasi berbasis android.
2. Pengembang aplikasi dapat melakukan pengembangan pada kode yang ada dengan lebih leluasa

1.4. Ruang Lingkup

Ruang lingkup skripsi ini dibatasi pada beberapa aspek, seperti :

1. Skripsi berfokus pada perubahan struktur perangkat lunak
2. Tidak dilakukan perbaikan *bug* pada kode yang sudah ada
3. Dilakukan pengujian JHawk setelah transformasi struktur perangkat lunak untuk mengetahui perbandingan kualitas perangkat lunak sebelum transformasi dan sesudah transformasi.