Implementasi dan Evaluasi Pengaruh MVVM *Pattern* terhadap *Reusability* pada Aplikasi Berbasis Mobile Android (Studi Kasus Sidang Tugas Akhir S1 Informatika Telkom University)

1st Fahry Fauzan Iskandar
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
fahryfauzan@students.telkomuniversity.ac.id

2nd Gede Agung Ary Wisudiawan
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
degunk@telkomuniversity.ac.id

3rd Shinta Yulia Puspitasari
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
shintayulia@telkomuniverity.ac.id

Abstrak-Perkembangan perangkat lunak meningkat seiring dengan banyaknya permintaan pengguna, namun wak-tu yang harus diselesaikan dalam pengembangan sangat terbatas. Salah satu solusi dari permasalahan tersebut adalah dengan menerapkan *Software Reuse* atau penggunaan kembali suatu perangkat lunak di-karenakan dapat meningkatkan kualitas dan mempercepat waktu yang dibutuhkan dalam pengembangan perangkat lunak. *Reusability* memungkinkan *source code* dapat digunakan kembali untuk menambahkan sedikit fungsionalitas atau tanpa modifikasi. Namun, permasalahan dalam *reusability* adalah jika aplikasi yang telah ada memiliki *code smell* seperti *coupling* dan kompleksitas yang tinggi pada suatu *class*. Sehingga berdampak pada sulitnya suatu code untuk didaur ulang. Oleh karena itu, maka perlu dilakukan *refactor* atau perubahan kode pada aplikasi. Pada studi kasus ini, aplikasi direfactor kedalam arsitektur MVVM karena memiliki keunggulan yaitu kohesi yang tinggi dan tingkat *coupling* yang rendah. Untuk penguji- an *reusability*, menggunakan pengukuran CK-metrics yang berkaitan dengan aspek *reusability* diantaranya yaitu CBO (Coupling Between Object), DIT (Depth of Inheritance Tree), NOC (Number of Children), WMC (Weighted Methods per Class), dan LCOM (Lack of Cohesion in Method). Setelah dilakukan refactoring pada aplikasi *baseline* dan dilakukan pengukuran terhadap metriks tersebut, arsitektur MVVM menu- runkan nilai *reusability* pada aplikasi. Hal itu disebabkan oleh meningkatnya *coupling* dan kompleksitas, serta menurunnya kohesi pada aplikasi.

Kata kunci - *Reusability*, Arsitektur MVVM, CK-metrics.

Abstract-Software development increases with the number of user requests, but time That which must be completedin development is very limited. One solution to the problem That is by implementing Software Reuse beca-use it can improve quality and speed up the time needed in development software. Reusability allows sourcecode to be reused to add little functionality or no modification. However, the problem with reusability is that if the application existing ones have a code smell like coupling and high complexity in a class. So that impact on the difficulty of a code to be reused. Therefore, it is necessary to refactor or code changes tothe application. In this case study, the application is refactored into the MVVM architecture because it hasthe advantages of high cohesion and low coupling rate. For testing reusability, using CK-metrics measu- rements related to reusability aspects including namely CBO (Coupling Between Objects), DIT (Depth of Inheritance Tree), NOC (Number of Children), WMC (Weighted Methods per Class), and LCOM (Lack ofCohesion in Method). After refactoring on the baseline application and measurements were made against these metrics, the MVVM architecture reduces the reusability value on the application. This is due to the increasing coupling and complexity, and decreased cohesion in the application.

Keywords- Reusability, MVVM architecture, CK-metrics.

I. PENDAHULUAN

A. Latar Belakang

Saat ini keragaman dan pengguna produk perangkat lunak meningkat. Sehingga waktu pengembangan produkterbatas. Salah satu solusi dari permasalahan tersebut adalah dengan menerapkan *Software Reuse* [22]. *Software Reuse* digunakan untuk meningkatkan kualitas dan produktivitas pengembangan perangkat lunak, sehingga da- pat membangun sistem yang lebih besar, lebih kompleks, lebih dapat diandalkan, biaya lebih murah, dan dapat diselesaikan dengan tepat waktu [6].

Reusability memungkinkan *source code* pada aplikasi dapat digunakan kembali untuk menambahkan sedikit fungsionalitas baru atau tanpa modifikasi [19]. Salah satu agar perangkat lunak memiliki kriteria *reusability* adalah dengan menggunakan *software architecture* [6]. Menurut shahbudin [18], Salah satu aspek yang harus diperhatikan dalam pengembangan aplikasi mobile adalah menggunakan pola arsitektur yang dapat meningkatkan *reusability* dan *maintainability*. *Architecture pattern* pada mobile diketahui ada tiga yaitu MVC, MVP, dan MVVM [21] [2]. Arsitektur yang digunakan untuk meningkatkan *reusability* pada penelitian ini adalah MVVM dengan keung- gulannya yaitu *high cohesion* dan *low coupling* yang memisahkan *User Interface* dan *logic* [11]. Selain itu, Ö ztürk melakukan perbandingan kualitas pada

aplikasi dengan arsitektur MVP dan MVVM, hasilnya MVVM memiliki coupling yang baik dikarenakan hanya lapisan *view* yang memiliki ketergantungan dengan lapisan *presentation*,
sedangkan MVP memiliki ketergantungan dua arah antara lapisan view dan presentation [15].

Pada penelitian ini, untuk mengukur *reusability* diusulkan CK-metrics yang terdiri dari WMC, NOC, DIT, LCOM, dan CBO [13]. Dari CK-metrics ditemukan bahwa kompleksitas, kohesi dan *coupling* dapat membantu dalam memprediksi nilai *reusability* dalam perangkat lunak [16].

Penelitian studi kasus ini akan membangun aplikasi tugas akhir S1 Informatika berbasis android berdasarkan aplikasi yang telah dibuat yaitu aplikasi tugas akhir D3 Informatika. Penerapan arsitektur MVVM pada aplikasi tersebut diharapkan dapat meningkatkan *reusability*. Selain itu tujuan dibangun aplikasi tugas akhir S1 Informatika ini adalah dapat digunakan program studi yang lain sehingga jika program studi yang lain ingin membuat aplikasiserupa tidak perlu dibangun dari awal.

B. Rumusan Masalah

Berikut rumusan masalah yang ingin diangkat adalah:

A. Bagaimana cara agar aplikasi dapat digunakan kembali?

B. Bagaimana cara mengimplementasi aplikasi kedalam arsitektur MVVM?

C. Bagaimana cara identifikasi komponen pada aplikasi Tugas Akhir D3 Informatika untuk digunakan kembalipada aplikasi Tugas Akhir S1 Informatika?

D. Bagaimana cara mengubah alur proses bisnis / fungsionalitas aplikasi Tugas Akhir D3 Informatika menjadiaplikasi Tugas Akhir S1 Informatika?

E. Bagaimana pengaruh arsitektur MVVM pada aplikasi terhadap nilai *reusability*?

C. Batasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. *Architecture pattern* yang digunakan pada penelitian ini adalah MVVM *pattern*.
2. Aplikasi yang dibangun sebagai studi kasus adalah Aplikasi Tugas Akhir S1 Informatika dengan mengambil sebagian komponen dari aplikasi Tugas Akhir D3 Informatika sebagai baseline aplikasinya.
3. Metrik untuk mengukur nilai Reusability adalah CK Metrics.

D. Tujuan

Berikut adalah tujuan yang ingin dicapai pada penulisan proposal/TA:

1. Mengetahui cara agar aplikasi dapat digunakan kembali.
2. Mengetahui cara mengimplementasi aplikasi kedalam arsitektur MVVM.
3. Mengetahui cara identifikasi komponen pada aplikasi Tugas Akhir D3 Informatika untuk digunakan kembali pada aplikasi Tugas Akhir S1 Informatika.
4. Mengetahui cara mengubah alur proses bisnis / fungsionalitas aplikasi Tugas Akhir D3 Informatika menjadi aplikasi Tugas Akhir S1 Informatika.
5. Mengetahui pengaruh arsitektur MVVM pada aplikasi berbasis mobile terhadap nilai *reusability*?

II. KAJIAN TEORI

Berikut adalah penelitian terkait yang telah dipublikasikan sebelumnya.

Wisnuadhi [24] dalam penelitiannya melakukan perbandingan performansi antara arsitektur MVP dan MVVMdengan melakukan test case sebanyak 9 kali, hasilnya MVVM memiliki perfomansi yang lebih baik dalam se-gi penggunaan CPU dengan rata-rata perbedaan 0.55%, dan waktu eksekusi dengan rata-rata perbedaan sebesar
126.21 ms dibandingkan MVP.

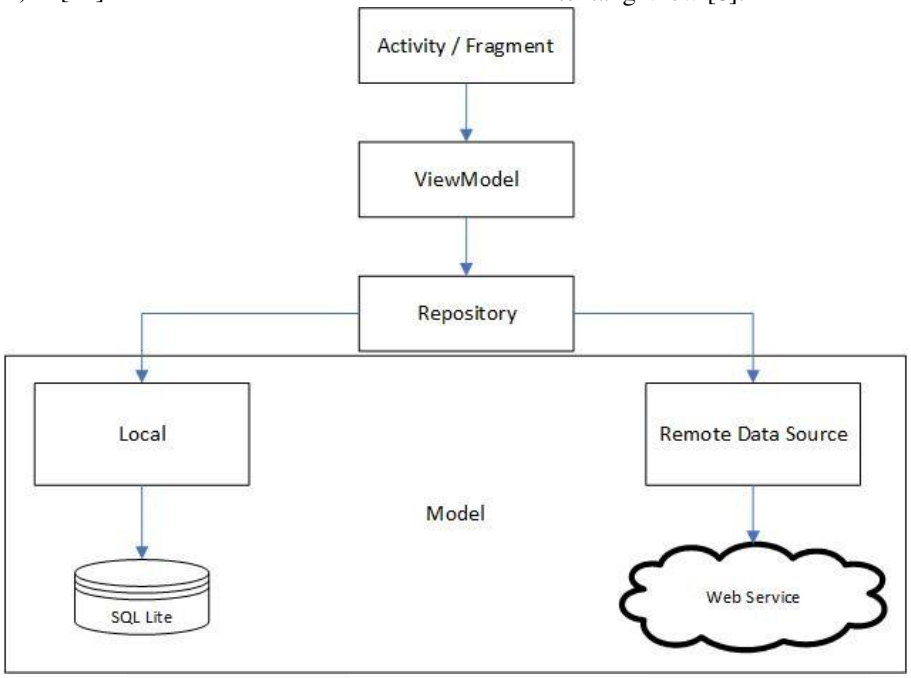
Lou [12] dalam penelitiannya melakukan perbandingan antara MVC, MVP, dan MVVM. Pengujian dilakukan dengan melakukan pengukuran terhadap tiga faktor kualitas yaitu testability, modifiablity, dan performance. Ha- silnya MVVM memiliki nilai testability yang lebih baik dari dua arsitektur lainnya karena kriteria dari testability yaitu *Size of test Cases* dan *Consumed time to run test cases* dengan nilai terkecil yang berarti lebih baik meskipun memiliki hasil terendah dari *ease to debug with break points* yang berarti lebih buruk. Untuk modifiability, arsi- tektur MVVM dan MVP lebih baik dari MVC. Untuk performance khususnya dalam konsumsi memori MVVM dan MVP memiliki konsumsi memori yang lebih rendah dari MVC.

Li [11] dalam penelitiannya menerapkan arsitektur MVVM pada MES (Manufacturing Execution System). Permasalahan pada MES adalah terdapat *requirement* yang berbeda pada setiap perusahaan manufaktur. Untuk mengurangi modifikasi pada kode dan mengurangi *coupling* antara halaman view dan data, penelitian ini meng- usulkan arsitektur MVVM karena memiliki atribut *high cohesion* dan *low coupling*. Hasilnya adalah MVVM dapat menyelesaikan permasalahan pada MES yang dalam pengembangannya harus disesuaikan dan mengurangi pengaruh modifikasi pada sistem karena perubahan

kebutuhan pada klien.

A. MVVM (Model-View-ViewModel)

Pada tahun 2005, Gossman John dari Microsoft memperkenalkan model arsitektur MVVM (Model-View- ViewModel) [20]. Arsitektur MVVM terinspirasi dari MVP dan MVC. Pada MVVM, ViewModel menggantikan peran presenter dan controller dimana tanggung jawab antara View dan ViewModel berbeda [9]. pada MVVM view mengetahui tentang perubahan pada Presentation Model, namun Presentation Model tidak mmengetahui tentang View [8].



GAMBAR 1.
STRUKTUR MODEL-VIEW-VIEWMODEL [23]

Berdasarkan struktur MVVM pada gambar 1, MVVM terbagi menjadi tiga bagian. Bagian pertama adalah view berupa activity/fragment yang bertanggung jawab untuk mengatur tampilan kepada *user*. View juga melakukan observasi kepada viewmodel untuk melihat perubahan data pada viewmodel sehingga view akan otomatis akan perubahan data jika adanya perubahan pada viewmodel. Bagian kedua yaitu ViewModel yang berperan sebagai jembatan antara view dan model dan memiliki tanggung jawab yaitu memperhatikan ketersediaan data dari API.

Bagian terakhir yaitu Model yang bertanggung jawab untuk menjalankan API dan mengatur kesediaan data dari repository. Repository dibagi menjadi 2 bagian yaitu local dimana data disimpan secara local seperti menggunakan SQL Lite, dan yang kedua dengan remote data source dimana data diambil dan disimpan pada web service [23].

B. Reusability

Reusability merupakan properti dari aset perangkat lunak yang menunjukkan kemungkinan perangkat lunak tersebut untuk dapat digunakan kembali. Penggunaan kembali suatu perangkat lunak yang ada disebut sebagai *Software Reuse* [6]. *Software Reuse* digunakan untuk membantu proses

pengembangan software yang dapat me- ningkatkan kualitas *software*, dan waktu penyelesaian pengembangan perangkat lunak yang cepat dengan meng- gunakan sedikit tenaga ahli, alat, dan metode sehingga effort dan biaya yang dibutuhkan dapat berkurang juga dapat menciptakan software dengan kualitas yang baik dengan meningkatkan integrasi pada system. Dengan de- mikian, *software reuse* bertujuan untuk menjadikan desain produk perangkat lunak, code, dan komponen lainnyaagar dapat didaur ulang sehingga dapat memotong biaya, waktu, dan meningkatkan kualitas produk [3].

Software reuse memiliki berbagai bentuk atau level dalam seperti System reuse, application reuse, component reuse, object dan function reuse. Untuk membangun aplikasi ini akan dibangun dengan pendekatan Object dan Function Reuse karena tidak semua class dan objek akan digunakan kembali dan komponen yang akan digunakan kembali adalahfungsi yang serupa dengan aplikasi baselinenya.

C. CK-Metrics

Pada tahun 1994, Chidamber dan Kemerer memperkenalkan standar metrik yang digunakan untuk mengukur kualitas perangkat lunak [4] [14]. Metriks tersebut disebut sebagai CK Metrics. Dalam CK Metrics ada 5 metrik untuk mengukur faktor kualitas *reusability* [13] yang terdiri dari:

1. Weighted Methods per Class (WMC)
   Metrik ini digunakan untuk menghitung jumlah kompleksitas semua method dalam suatu kelas. Kelas de- ngan jumlah method yang besar dapat membatasi kemungkinan aplikasi dapat didaur ulang karena menan- dakan aplikasi menjadi lebih spesifik sehingga kompleksitas menjadi tinggi. Sehingga, semakin tinggi nilai WMC maka semakin rendah nilai *reusability*. WMC memiliki persamaan sebagai berikut.

$$\text{WMC} = \sum_{I=1}^{n} c_i \qquad (1)$$

   dimana:
   n = jumlah method dalam suatu kelas ci = kompleksitas dari sebuah method

   Untuk menghitung kompleksitas dalam suatu method salah satunya adalah dengan cyclomatic complexity (CC) [10].

2. Depth of Inheritance Tree (DIT)
   Metrik ini digunakan untuk menghitung berapa panjang maksimum jalur pewarisan (*inheritance*) dari no- de ke root. Semakin dalam kelas dalam hierarki, maka potensi penggunaan kembali pada metode yang diwariskan semakin besar. Oleh karena itu, Semakin tinggi nilai DIT maka semakin tinggi nilai *reusability*.

3. Number of Children (NOC)
   Metrik ini digunakan untuk menghitung jumlah subclass atau jumlah pewarisan langsung dalam sebuah kelas. Metrik ini mengukur berapa banyak subclass yang akan mewarisi method dari kelas induk. Semakin tinggi nilai NOC maka semakin tinggi nilai *reusability*.

4. Coupling between object classes (CBO)
   Metrik ini digunakan untuk menghitung jumlah kelas yang saling memiliki ketergantungan. Kelas bisa disebut memiliki ketergantungan dengan kelas lain jika dalam suatu kelas tersebut menggunakan method atau variabel dari kelas yang lain. Semakin banyaknya kelas yang memiliki ketergantungan, maka semakin sulit suatu komponen untuk digunakan kembali pada aplikasi yang lain. Selain itu, melakukan perubahan pun semakin sulit karena semakin tinggi sensitivitas terhadap perubahan pada kelas lainnya. Sehingga, semakin tinggi nilai CBO maka semakin rendah nilai *reusability*.

5. Lack of Cohesion in Methods (LCOM)
   Metrik ini digunakan untuk menghitung jumlah metode yang berbeda dalam suatu kelas yang mereferen- sikan instance variable yang diberikan. Semakin tinggi nilai LCOM maka semakin rendah nilai *reusability*karena semakin rendah cohesion maka semakin tinggi kompleksitasnya sehingga meningkatnya kesalahan selama proses pengembangan. Untuk menghitung LCOM, dapat menggunakan persamaan dibawah ini [7].

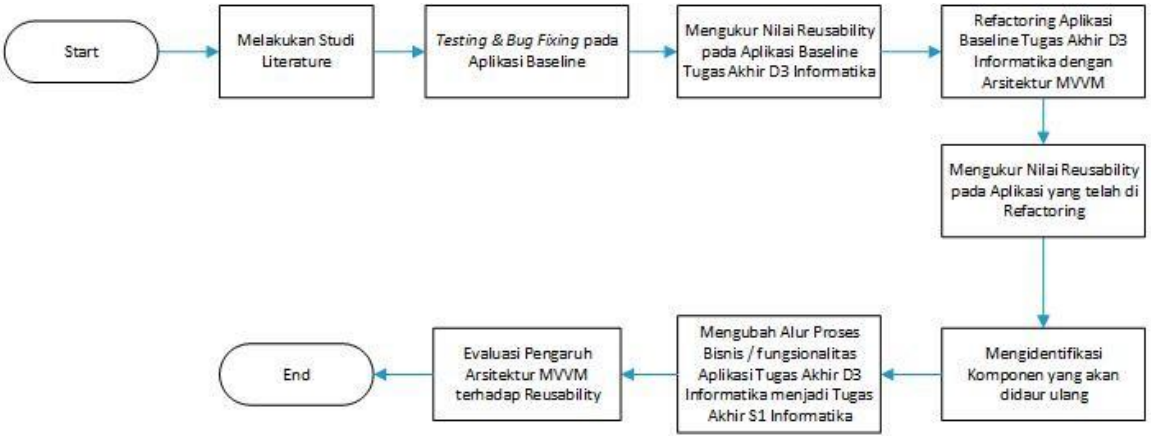
D. CodeMR

CodeMR merupakan salah satu tool yang digunakan untuk mengukur kualitas perangkat lunak. CodeMR memvisualisasikan kode metrik dan atribut kualitas tinggi seperti *coupling*, kompleksitas, dan ukuran. CodeMR mendukung perhitungan metriks seperti WMC, DIT, NOC, CBO, dan LCOM yang digunakan untuk mengukur *reusability* suatu perangkat lunak [1].

$$\text{LCOM3} = \frac{(m - sum(mA)/a)}{(m-1)} \qquad (2)$$

dimana:
m = jumlah method dalam suatu kelas.
sum(mA) = jumlah mA dalam suatu kelas.
mA = jumlah method yang mengakses variabel.
a = jumlah variabel dalam suatu kelas baik yang statis maupun tidak.

III. METODE



GAMBAR 2.
ALUR METODE PENELITIAN

Berdasarkan flowchart alur metode penelitian pada gambar 2 berikut penjelasan pada setiap tahapannya :

A. Melakukan Studi Literature

Pada tahap ini, dilakukan pengumpulan bahan berupa jurnal penelitian sebelumnya untuk studi literature serta file aplikasi Tugas Akhir D3 Informatika sebagai baseline untuk membangun aplikasi sidang Tugas Akhir S1 Informatika. Beberapa hal yang dipelajari dari penelitian sebelumnya adalah sebagai berikut:

1. Kajian pustaka terkait arsitektur MVVM.
2. Kajian pustaka terkait *reusability* pada perangkat lunak.
3. Kajian pustaka terkait CK-metrics.
4. Mempelajari aplikasi Tugas Akhir D3 Informatika.

Hasil yang didapatkan dari mempelajari penelitian sebelumnya adalah dapat memahami arsitektur MVVM, *Software Reuse*, dan CK Metrics sebagai metrik untuk mengukur nilai *reusability* pada perangkat lunak. Selain itu, dari mempelajari aplikasi baseline adalah dengan mengetahui user/entitas, tugas dan fungsi dari user, dan pro-ses bisnis dari aplikasi tugas akhir D3 Informatika. Setelahnya didapatkan fungsi mana saja yang harus diterapkan pada sidang Tugas Akhir S1. Fungsi-fungsi tersebut telah didefinisikan sebelumnya pada penelitian yang dilakuk-an oleh prayogo [17]. Berikut tabel 1 yang memperlihatkan fungsi/fitur yang dapat atau tidak dapat digunakan kembali.

TABEL 1.
DAFTAR FUNGSI/FITUR PADA APLIKASI TUGAS AKHIR D3 INFORMATIKA YANG DIGUNAKAN ATAU TIDAK DIGUNAKANKEMBALI [17].

Users	Fungsi/Fitur	Deskripsi	Status
Koordinator Proyek Akhir	Mengolah data informasi	Fungsi ini digunakan untuk mengolah informasi terkait proses pelaksahaan proyek akhir D3	Digunakan kembali
	Mengolah data data dosen	Fungsi ini digunakan untuk mengolah data dosen tetap D3	Digunakan kembali
	Mengolah data mahasiswa	Fungsi ini digunakan untuk mengolah data mahasiswa D3 yang melaksanakan proyek akhir	Digunakan kembali

	Mengubah profil	Fungsi ini digunakan untuk mengubah profil koordinator proyek akhir	Digunakan kembali
	Menambahkan kategori judul	Fungsi ini digunakan untuk menambahkan kategori judul yang akan digunakan untuk menambahkan judul oleh dosen.	Tidak digunakan kembali
	Menambahkan kategori monitoring dan evaluasi	Fungsi ini digunakan untuk menambahkan kategori monitoring dan evaluasi yang akan digunakan untuk menambahkan monitoring dan evaluasi.	Tidak digunakan kembali
	Menambahkan judul dosen	Fungsi ini digunakan untuk menambahkan judul untuk dosen bila dosen yang bersangkutan ingin dibuatkan judul oleh koordinator Proyek Akhir.	Tidak digunakan kembali
	Pemetaan monitoring dan evaluasi	Fungsi ini digunakan untuk memetakkan kelompok proyek akhir dengan dosen reviewer untuk melaksanakan monitoring dan evaluasi.	Tidak digunakan kembali
Dosen	Mengolah Informasi	Fungsi ini digunakan untuk mengolah informasi terkait proses pelaksahaan proyek akhir D3	Digunakan kembali
	Menyetujui bimbingan	Fungsi ini digunakan dosen untuk menyetujui dengan hasil input bimbingan yang dilakukan oleh mahasiswa.	Digunakan kembali
	Mengubah Profil	Fungsi ini digunakan untuk mengubah profil dosen	Digunakan kembali
	Menambahkan judul	fungsi ini digunakan untuk menambahkan judul untuk digunakan oleh mahasiswa sebagai judul proyek akhir.	Tidak digunakan kembali
	Menyetujui dan menolak judul dosen	Fungsi ini digunakan untuk menerima ataupun menolak judul yang ajukan oleh kelompok berdasarkan judul yang diajukan dosen.	Tidak digunakan kembali

	Menyetujui dan menolak judul mandiri	Fungsi ini digunakan untuk menerima ataupun menolak judul yang ajukan oleh kelompok mahasiswa berdasarkan judul yang diajukan secara mandiri.	Tidak digunakan kembali
	Menambahkan nilai dan review monitoring dan evaluasi	Fungsi ini digunakan dosen untuk menambahkan hasil monitoring dan evaluasi untuk masing–masing mahasiswa.	Tidak digunakan kembali
	Menambahkan nilai sidang	Fungsi ini digunakan dosen untuk menambahkan nilai dan review sidang untuk masing-masing mahasiswa.	Digunakan kembali
Mahasiswa	Menambahkan bimbingan	Fungsi ini digunakan oleh mahasiswa untuk menambahkan bimbingan dan setelah menambahkan bimbingan,hasil nya harus disetujui terlebih dahulu oleh dosen pembimbing.	Digunakan kembali
	Mengubah Profil	Fungsi ini digunakan untuk mengubah profil mahasiswa	Digunakan kembali
	Mengajukan judul dosen	Fungsi ini digunakan oleh mahasiswa untuk memilih judul dari dosen.	Tidak digunakan kembali
	Mengajukan judul mandiri	Fungsi ini digunakan oleh mahasiswa untuk mengajukan judul secara mandiri kepada dosen yang ingin dijadikan pembimbing.	Tidak digunakan kembali

Tabel 1 juga merupakan acuan untuk dapat mengidentifikasi *class* yang akan digunakan kembali. Adapun fung- si yang tidak terdapat pada aplikasi baseline D3 Informatika namun ada pada aplikasi tugas akhir S1 Informatika. *Requirement* untuk kebutuhan aplikasi tugas akhir S1 Informatika diuraikan pada tabel 2 berikut.

TABEL 2.
FUNCTIONAL REQUIREMENT UNTUK TUGAS AKHIR S1 INFORMATIKA [17].

Users	Functional Requirement
Mahasiswa	Mahasiswa dapat melihat status SKTA
	Mahasiswa dapat mengajukan form SKTA
	Mahasiswa dapat melakukan bimbingan
	Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur terjadwal
	Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur reguler
	Mahasiswa dapat melihat jadwal sidang terjadwal
	Mahasiswa dapat mengumpulkan lembar revisi
Dosen	Dosen dapat memvalidasi Bimbingan
	Dosen dapat memvalidasi form pendaftaran sidang reguler
	Dosen pembimbing dan penguji dapat memberikan nilai sidang
	Dosen dapat memvalidasi form SK
Prodi	Admin prodi dapat mengunggah form excel plotting pembimbing dan mahasiswa serta membuat SK TA
	Admin prodi dapat melihat daftar mahasiswa dan dosen pembimbing
	Admin prodi dapat mempublish excel terkait jadwal sidang
	Admin prodi melihat daftar nilai dan berita acara sidang
	Admin prodi dan LAAK melihat daftar status revisi peserta sidang
	Admin prodi dapat mengelola data mahasiswa
	Admin prodi dapat mengelola data dosen
LAAK	Admin LAAK dapat mengatur periode pendaftaran sidang
	Admin LAAK dapat melihat daftar mahasiswa peserta sidang

B. *Testing & Bug Fixing* pada Aplikasi Baseline

Pada tahapan ini, dilakukan testing pada aplikasi untuk memastikan ada atau tidaknya bug pada aplikasi. Tes-ting dilakukan dengan mencoba semua fungsionalitas pada aplikasi seperti *create, read, update*, dan *delete* data. Setelah dilakukan pengecekan, ada beberapa bug pada fungsionalitas, salah satunya adalah fitur tambah nilai si- dang dimana setelah fungsionalitas tersebut dijalankan, data tidak bertambah pada aplikasi dan database. Setelah dilakukan penelusuran, ditemukan penyebab bug tersebut adalah kode pada route API di web servicenya. Berikut kode sebelum diperbaiki dan sesudah diperbaiki.

```
Route::resource('sidang', 'ApiController\ApiControllerSidang',
```

GAMBAR 3.
KODE SEBELUM DIPERBAIKI

```
Route::resource('sidang', ApiControllerSidang::class,
```

GAMBAR 4.
KODE SETELAH DIPERBAIKI

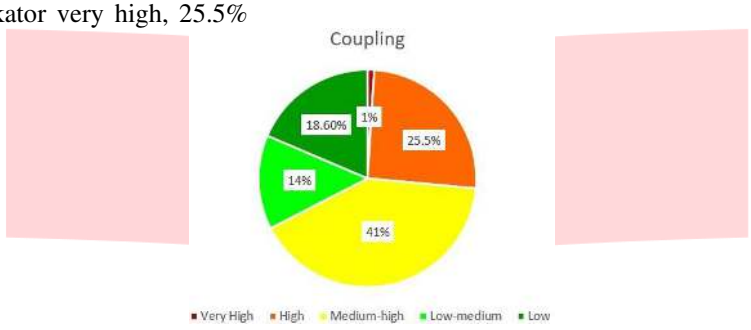
C. Mengukur Nilai Reusability pada Aplikasi Baseline Tugas Akhir D3 Informatika

Dalam tahapan ini, proses yang dilakukan adalah mengukur nilai *reusability* pada aplikasi. Pengukuran nilai *reusability* adalah dengan menggunakan CK

Metrics berdasarkan studi literature sebelumnya. Proses pengukur- an dilakukan dengan memasukkan semua *class* pada aplikasi ke dalam *tool* CodeMR. Setelah proses pengukuran selesai, maka codeMR menampilkan nama *class* dengan nilai metrik yaitu CBO, DIT, NOC, WMC, dan LCOM. Setelah pengukuran dilakukan, terdapat class-class yang bermasalah karena memiliki *high coupling*. Berikut me- rupakan tabel class yang bermasalah pada Lampiran 1 tabel 10.

Untuk melihat persentase *class* yang bermasalah diperlihatkan pada gambar 5 berikut ini.

Pada gambar 5 terdapat 1% class yang memiliki tingkat *coupling* dengan indikator very high, 25.5% class memiliki tingkat *coupling* dengan indikator *High* dan 41% dengan tingkat indikator *Medium-high*. Akibatnya se-makin banyaknya *class* yang memiliki tingkat *coupling* yang tinggi, maka semakin banyak class yang memiliki ketergantungan dengan class lainnya sehingga jika ada perubahan pada class tersebut maka akan berdampak padaclass lainnya dan melakukan perubahan membutuhkan lebih banyak usaha. Hal itu juga mengakibatkan penggu- naan kembali suatu aplikasi sulit untuk dilakukan. Diperlihatkan juga hasil rata-rata nilai metrik pada semua *class*pada tabel 3 berikut.



GAMBAR 5.
PERSENTASE COUPLING PADA *CLASS*

TABEL 3.
PENGUKURAN CK METRICS PADA APLIKASI BASELINE TUGAS AKHIR D3 INFORMATIKA SEBELUM REFACTORING.

Aplikasi Baseline TA D3 Informatika	CK Metrics				
	CBO	DIT	NOC	WMC	LCOM
	10.73	3.87	0.59	10.18	0.50

D. *Refactoring* Aplikasi Baseline dengan Arsitektur MVVM

Selanjutnya, proses yang dilakukan adalah melakukan *refactoring* aplikasi baseline menjadi arsitektur MVVM. *Refactoring* bertujuan untuk meningkatkan kualitas pada kode dengan mengubah struktur internal pada komponenperangkat lunak agar mudah dipahami dan dimodifikasi tanpa mengubah fungsionalitas pada perangkat lunak. *Refactoring* dapat mempengaruhi kualitas internal seperti *complexity*, *coupling*, dan *inheritance* serta kualitas eksternal seperti *maintainability, reusability, understandability, dan efficiency* [5]. Oleh karena itu, *refactoring* dapat meningkatkan *software reuse*.

Langkah pertama sebelum *refactoring* adalah dengan menganalisa *class diagram* pada aplikasi baseline. *Classdiagram* dari aplikasi baseline adalah pada gambar 26. Berdasarkan gambar 26 *class diagram*, terdapat *class* DosenSidangTambahActivity yang merupakan bagian dari view pada android. DosenSidangTambahActivity ter- hubung ke SidangPresenter dan ProyekAkhirPresenter yang merupakan penghubung antara *view* dan model. Si-dangPresenter dan ProyekAkhirPresenter terhubung pada *class* ApiService yang merupakan model untuk menye- diakan sumber data. SidangWorkView dan ProyekAkhirWorkView merupakan *class interface* yang berisi *method*yang akan digunakan oleh *view*, Dalam hal ini view melakukan *overidding* atau mengimplementasikan *method* pada class interface. Berdasarkan analisa, maka aplikasi baseline menggunakan arsitektur MVP (Model-View-Presenter).

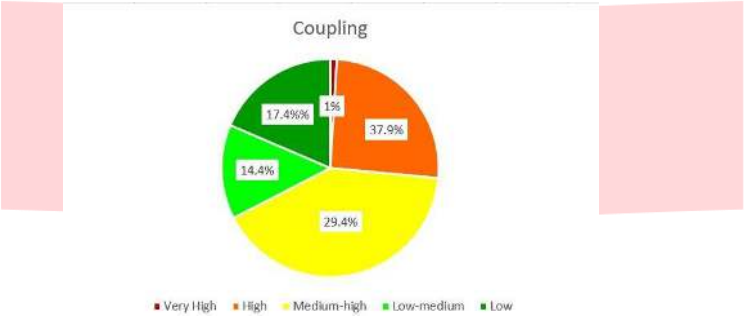
Berikutnya adalah mengubah struktur aplikasi kedalam bentuk arsitektur MVVM. Cara implementasi arsitek- tur MVVM adalah dengan mengikuti struktur seperti pada gambar 1. Hasil struktur aplikasi yang telah diubah kedalam bentuk arsitektur MVVM adalah pada gambar 28. Pada gambar 28, DosenSidangTambahActivity ber- peran sebagai *view* dan terhubung pada dua *class* yaitu SidangViewModel dan ProyekAkhir ViewModel. Sida- ngViewModel dan ProyekAkhir ViewModel berperan untuk menyimpan data berupa LiveData agar *view* dapat melakukan observasi terhadap ViewModel, ViewModel juga terhubung dengan repository (SidangRepository danProyekAkhirRepository) yang juga bertanggung jawab sebagai model untuk mendapatkan data, dan melakukan perubahan data. Repository juga bertanggung jawab untuk mengatur sumber data yang dibutuhkan oleh aplikasi baik itu secara local maupun remote seperti yang ditunjukan pada gambar 3. Pada aplikasi ini sumber data yang di-ambil berasal secara remote dari server dalam bentuk

RestAPI. Perbedaan dari struktur pada gambar 2 sebelumnya, tidak terdapat *class interface* yang terhubung ke view. Implementasi MVVM dituliskan dalam bentuk *code* pada gambar 18 *class* DosenSidangTambahActivity, 19 *class* SidangViewModel, 20 *class* ProyekAkhirViewModel, 21 *class* SidangRepository, dan 22 *class* ProyekAkhirRepository.

E. Mengukur Nilai Reusability pada Aplikasi yang telah di Refactoring

Dalam tahapan ini, proses yang dilakukan adalah mengukur kembali aplikasi yang telah direfactoring pada tahapan sebelumnya. Pengukuran dilakukan bertujuan untuk memastikan apakah nilai *coupling* mengalami penu-runan atau tidak. Hasil persentase pengukuran adalah terdapat pada gambar 6 berikut.

Pada gambar 6, terdapat 1% class dengan indikator very-high, 37.9% high, 29.4% medium-high, 14.4% low- med, dan 17.4% low. Berdasarkan perbandingan terhadap pengukuran sebelum direfactoring, class yang memilikiindikator high mengalami peningkatan 12.4%. Untuk lebih detailnya, berikut tabel 11 yang terdapat pada lampiran



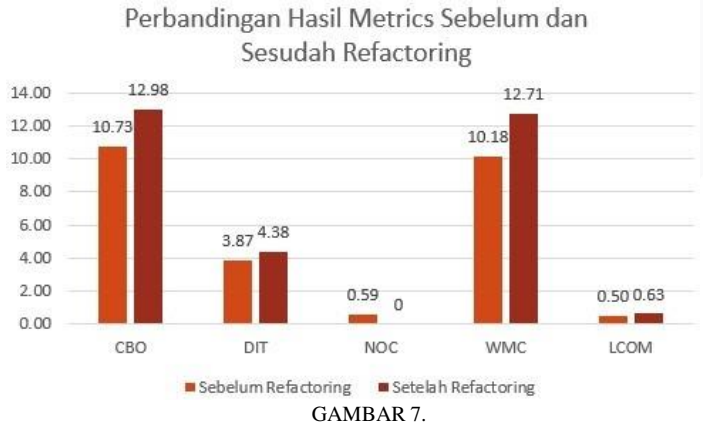
GAMBAR 6.
PERSENTASE COUPLING PADA *CLASS* SETELAH REFACTORING

2 dengan nilai *coupling* beserta indikatornya. Hasil rata-rata nilai metrics dari semua class pada aplikasi baselineyang terdapat pada tabel 4 berikut.

TABEL 4.
PENGUKURAN CK METRICS PADA APLIKASI BASELINE TUGAS AKHIR D3 INFORMATIKA SETELAH DI REFACTORING.

Aplikasi Baseline TA D3 Setelah Refactoring (MVVM)	CK Metrics				
	CBO	DIT	NOC	WMC	LCOM
	12.98	4.38	0	12.71	0.63

Hasil perbandingan nilai metriks antara sebelum dan sesudah refactoring adalah pada gambar 7 berikut.



GAMBAR 7.
PERBANDINGAN RATA-RATA NILAI METRIKS PADA *CLASS*

Untuk rata-rata nilai CBO mengalami peningkatan sebesar 2.25, DIT mengalami peningkatan sebesar 0.51, NOC mengalami penurunan sehingga nilainya menjadi 0, WMC mengalami peningkatan sebesar 2.53, dan LCOMmengalami peningkatan sebesar 0.13.

F. Mengidentifikasi Komponen yang akan didaur ulang (*Code Reuse*)

Dalam tahapan ini, proses yang dilakukan adalah mengidentifikasi dan memilih komponen-komponen pada aplikasi tugas akhir D3 Informatika yang akan digunakan kembali pada aplikasi sidang tugas akhir S1 Informatika. Pemilihan komponen didasarkan pada tahap studi literature sebelumnya. Komponen yang

diambil berupa *class* utuh atau hanya *fragment code* seperti *function* atau *method*, atribut, dan parameter pada suatu *class*. Pemilihan komponen juga ditentukan dari use case diagram pada gambar 17 dimana terdapat aktor yaitu prodi, mahasiswa, dan, dosen. Untuk use casenya terdiri dari mengolah data mahasiswa, mengolah data dosen, mengolah informasi,melihat data tugas akhir, membuat jadwal sidang, melihat absensi bimbingan, menambahkan absensi bimbingan, menyetujui bimbingan, melihat jadwal sidang penguji, melihat nilai sidang, dan mengelola nilai sidang mahasiswa.Setiap aktor harus login terlebih dahulu untuk dapat mengakses use case tersebut. Class yang akan digunakan kembali diperlihatkan pada tabel 5 berikut.

TABEL 5.
CLASS YANG AKAN DIGUNAKAN KEMBALI.

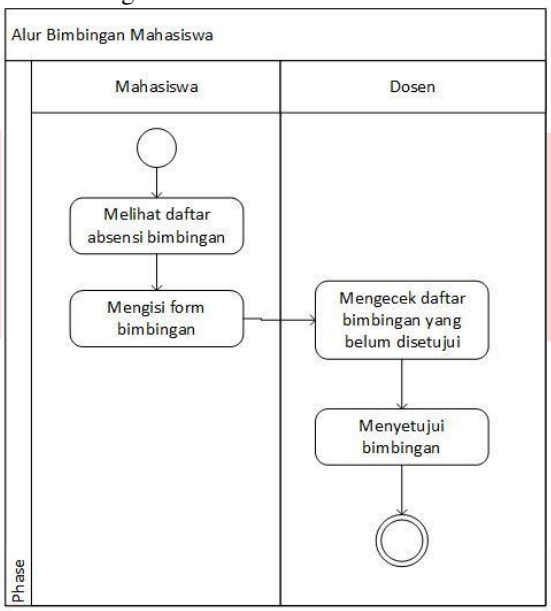
No	Daftar class yang digunakan kembali		
1	LoginActivity	40	DosenPaFragment
2	KoorMainActivity	41	KoorDosenFragment
3	MahasiswaMainActivity	42	KoorInformasiFragment
4	DosenMainActivity	43	KoorMahasiswaFragment
5	DosenBimbinganDetailActivity	44	KoorProyekAkhirFragment
6	DosenInformasiDetailActivity	45	MahasiswaInformasiFragment
7	DosenInformasiTambahActivity	46	MahasiswaPaFragment
8	DosenSidangTambahActivity	47	DosenBimbinganViewAdapter
9	DosenInformasiUbahActivity	48	DosenInformasiViewAdapter
10	DosenProfilUbahActivity	49	DosenPemberitahuanViewAdapter
11	DosenSidangUbahActivity	50	DosenProyekAkhirBimbinganViewAdapter
12	DosenPemberitahuanActivity	51	KoorDosenViewAdapter
13	DosenProfilActivity	52	KoorInformasiViewAdapter
14	DosenProyekAkhirActivity	53	KoorMahasiswaViewAdapter
15	DosenProyekAkhirBimbinganActivity	54	KoorPemberitahuanViewAdapter
16	DosenSidangActivity	55	KoorProyekAkhirViewAdapter
17	KoorDosenDetailActivity	56	MahasiswaInformasiViewAdapter
18	KoorInformasiDetailActivity	57	MahasiswaPaBimbinganViewAdapter
19	KoorMahasiswaDetailActivity	58	MahasiswaPemberitahuanViewAdapter
20	KoorProyekAkhirDetailActivity	59	Bimbingan
21	KoorDosenTambahActivity	60	Dosen
22	KoorInformasiTambahActivity	61	Informasi
23	KoorMahasiswaTambahActivity	62	KoordinatorPa
24	KoorDosenUbahActivity	63	Mahasiswa
25	KoorInformasiUbahActivity	64	Notifikasi
26	KoorMahasiswaUbahActivity	65	ProyekAkhir
27	KoorPemberitahuanActivity	66	User
28	KoorProfilUbahActivity	67	Sidang
29	MahasiswaBimbinganDetailActivity	68	ConnectionHelper
30	MahasiswaInformasiDetailActivity	69	MethodHelper
31	MahasiswaPaSidangDetailActivity	70	SessionManager
32	MahasiswaPaBimbinganTambahActivity	71	SpinnerHelper
33	MahasiswaPaBimbinganUbahActivity	72	FinproFirebaseMessagingService
34	MahasiswaProfilUbahActivity	73	FirebaseActivity
35	MahasiswaPaBimbinganActivity	74	ApiClient
36	MahasiswaPemberitahuanActivity	75	ApiService
37	MahasiswaProfilActivity	76	ApiUrl
38	DosenPaBimbinganFragment	77	Constant
39	DosenInformasiFragment		

Pada tabel 5 terdapat 77 dari 199 *class* yang digunakan kembali. Salah satu contoh komponen yang akan digunakan kembali adalah DosenSidangTambahActivity.java yang merupakan

class untuk menambahkan nilai si-dang pada mahasiswa. Namun ada beberapa atribut dan parameter yang harus diubah sehingga disesuaikan denganpenilaian sidang untuk S1 Informatika.

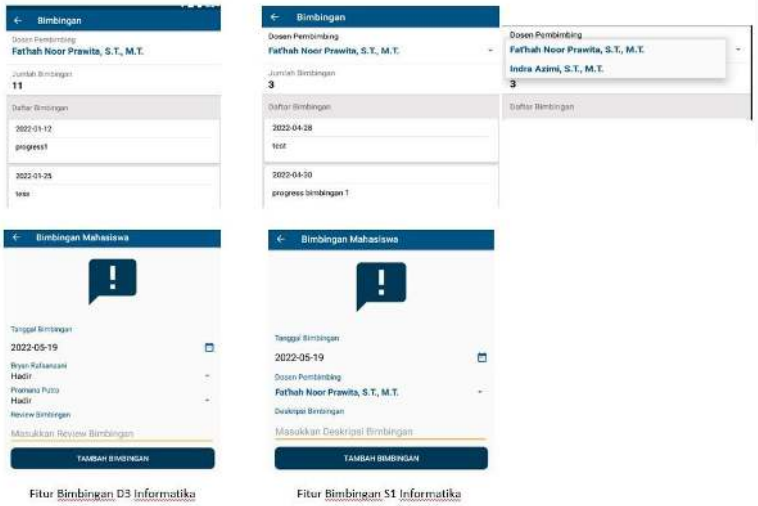
G. Mengubah Alur Proses Bisnis / fungsionalitas Aplikasi Tugas Akhir D3 Informatika menjadi Tu-gas Akhir S1 Informatika

Dalam tahapan ini, proses yang adalah mengubah alur proses bisnis / fungsionalitas aplikasi tugas akhir D3 In-formatika menjadi aplikasi sidang tugas akhir S1 Informatika. Komponen yang telah dipilih sebelumnya diterapk- an kembali pada proses pembangunan aplikasi sidang tugas akhir S1 Informatika. *Requirement* dalam membangunaplikasi telah didefinisikan pada studi literature. salah satu contoh dari *activity diagram* untuk modul bimbingan. *Activity diagram* digambarkan pada gambar 8 berikut.



GAMBAR 8.
ALUR PROSES BISNIS MODUL BIMBINGAN

Alur proses bisnis untuk modul bimbingan tidak mengalami perubahan, namun terdapat perbedaan dari sisi fungsionalitas. Pada aplikasi baseline (D3 Informatika), mahasiswa hanya melakukan bimbingan dengan satu dosen pembimbing sedangkan pada aplikasi S1 Informatika, mahasiswa dapat melakukan bimbingan dengan 1 atau 2 dosen pembimbing. Perbedaan fungsionalitas digambarkan pada gambar 9 berikut.



GAMBAR 9.
PERBEDAAN FUNGSIONALITAS APLIKASI D3 DAN S1 INFORMATIKA UNTUK MODUL BIMBINGAN

IV. HASIL DAN PEMBAHASAN

Berdasarkan hasil pengukuran pada aplikasi *baseline* sebelum dan setelah *refactor*. Hasilnya adalah dengan menerapkan arsitektur MVVM tidak meningkatkan *reusability* pada aplikasi, namun membuat *reusability* pada aplikasi menurun. Hal itu terlihat pada gambar 7 dimana terjadinya peningkatan pada nilai CBO, WMC, dan LCOM. Untuk lebih jelasnya, dipilih dua modul sampel yaitu modul tambah bimbingan dan modul tambah nilai sidang untuk D3 Informatika. Nilai CK-metriks sebelum dan sesudah refactoring untuk kedua modul adalah pada tabel 6 dan 7 berikut.

TABEL 6.
HASIL UKUR NILAI CK-METRICS SEBELUM REFACTORING UNTUK MODUL TAMBAH BIMBINGAN DAN TAMBAH NILAISIDANG D3 INFORMATIKA.

Module Name	Class Name	CK Metrics				
		CBO	DIT	NOC	WMC	LCOM
Tambah Bimbingan	MahasiswaPaBimbinganTambahActivity (View)	22	9	0	11	1.083
	BimbinganPresenter (Presenter)	13	1	0	29	0.722
	ICreate (View Interface)	0	1	6	4	0
Rata-rata		11.6	3,7	2	14.7	0.601
Tambah Nilai Sidang	DosenSidangTambahActivity (View)	22	9	0	13	0.942
	ProyekAkhirPresenter (Presenter)	11	1	0	28	0.575
	SidangPresenter (Presenter)	11	1	0	18	0.6
	ProyekAkhirWorkView (View Interface)	0	1	6	4	0
	SidangWorkView (View Interface)	0	1	2	4	0
Rata-rata		8.8	2.6	1.6	13.4	0.423

TABEL 7.
HASIL UKUR NILAI CK-METRICS SETELAH REFACTORING UNTUK MODUL TAMBAH BIMBINGAN DAN TAMBAH NILAISIDANG D3 INFORMATIKA.

Module Name	Class Name	CK Metrics				
		CBO	DIT	NOC	WMC	LCOM
Tambah Bimbingan	MahasiswaPaBimbinganTambahActivity (View)	24	9	0	14	0.943
	BimbinganViewModel (ViewModel)	11	2	0	29	0.989
	BimbinganRepository	4	1	0	7	0
Rata-rata		13	4	0	16.7	0.644
Tambah Nilai Sidang	DosenSidangTambahActivity (View)	23	9	0	18	0.956
	SidangViewModel (ViewModel)	8	2	0	14	1.042
	ProyekAkhirViewModel (ViewModel)	8	2	0	33	1.029
	SidangRepository	4	1	0	4	0
	ProyekAkhirRepository	4	1	0	9	0
Rata-rata		9.4	3	0	15.6	0.605

Berdasarkan hasil perhitungan dari kedua modul tersebut (Tambah Bimbingan D3 Informatika & Tambah Nilai Sidang D3 Informatika), perbandingan rata-rata nilai CK-metrics untuk sebelum dan sesudah refactoring diperlihatkan pada gambar 10 dan 11 berikut.

GAMBAR 10.
PERBANDINGAN RATA-RATA NILAI CK-METRICS PADA MODUL TAMBAH BIMBINGAN D3 INFORMATIKA SEBELUMDAN SETELAH REFACTORING

GAMBAR 11.
PERBANDINGAN RATA-RATA NILAI CK-METRICS PADA MODUL TAMBAH BIMBINGAN D3 INFORMATIKA SEBELUMDAN SETELAH REFACTORING

Sebagai tambahan, dilakukan juga perhitungan nilai ck-metrics dengan sampelnya yaitu modul tambah bim-bingan dan modul tambah nilai sidang untuk S1 Informatika pada tabel 8 dan 9 berikut.

TABEL 8.
HASIL UKUR NILAI CK-METRICS SEBELUM REFACTORING UNTUK MODUL TAMBAH BIMBINGAN DAN TAMBAH NILAISIDANG S1 INFORMATIKA.

Module Name	Class Name	CK Metrics				
		CBO	DIT	NOC	WMC	LCOM
Tambah Bimbingan	MahasiswaPaBimbinganTambahActivity (View)	23	9	0	12	0.917
	BimbinganPresenter (Presenter)	13	1	0	29	0.722
	PlottingPresenter (Presenter)	11	1	0	7	0.8
	ICreate (View Interface)	0	1	6	4	0
	PlottingListView (View Interface)	0	1	2	5	0
Rata-rata		9.4	2.6	1.6	11.4	0.488
Tambah Nilai Sidang	DosenSidangTambahActivity (View)	21	9	0	106	0.94
	SidangPresenter (Presenter)	11	1	0	12	0.675
	SidangWorkView (View Interface)	0	1	2	4	0
Rata-rata		10.6	3.7	0.7	40.7	0.538

TABEL 9.
HASIL UKUR NILAI CK-METRICS SETELAH REFACTORING UNTUK MODUL TAMBAH BIMBINGAN DAN TAMBAH NILAISIDANG S1 INFORMATIKA.

Module Name	Class Name	CK Metrics				
		CBO	DIT	NOC	WMC	LCOM
Tambah Bimbingan	MahasiswaPaBimbinganTambahActivity (View)	23	9	0	14	0.95
	BimbinganViewModel (ViewModel)	11	2	0	28	0.97
	PlottingViewModel (ViewModel)	8	2	0	15	1.029
	BimbinganRepository	4	1	0	6	0
	PlottingRepository	4	1	0	4	0
Rata-rata		10	3	0	13.4	0.589
Tambah Nilai Sidang	DosenSidangTambahActivity (View)	22	9	0	109	1.009
	SidangViewModel (ViewModel)	8	2	0	14	1.042
	SidangRepository	4	1	0	3	0
Rata-rata		11.3	4	0	42	0.683

Berdasarkan hasil perhitungan dari kedua modul tersebut (Tambah Bimbingan S1 Informatika & Tambah NilaiSidang S1 Informatika), perbandingan rata-rata nilai CK-metrics untuk sebelum dan sesudah refactoring diperli- hatkan pada gambar 12 dan 13

berikut.

GAMBAR 12.
PERBANDINGAN RATA-RATA NILAI CK-METRICS PADA MODUL TAMBAH BIMBINGAN S1 INFORMATIKA SEBELUMDAN SESUDAH REFACTORING

GAMBAR 13.
PERBANDINGAN RATA-RATA NILAI CK-METRICS PADA MODUL TAMBAH NILAI SIDANG S1 SEBELUM DAN SESUDAHREFACTORING

Berdasarkan perbandingan rata-rata untuk setiap atribut ck-metrics pada gambar 10 dan 11, rata-rata dari nilai CBO, DIT, WMC, dan LCOM mengalami kenaikan dan NOC mengalami penurunan setelah *refactoring* kedalam arsitektur MVVM pada kedua modul tersebut. Meningkatnya nilai CBO, WMC, LCOM dan turunnya nilai NOC menyebabkan penurunan kualitas perangkat lunak dalam aspek *reusability*. Berikut faktor perubahan nilai metricspada setiap atributnya pada tabel 10 berikut

TABEL 10. FAKTOR PERUBAHAN NILAI DARI SETIAP ATRIBUT CK-METRICS

.

<table>
<tr><td></td><td colspan="3">Ck-metrics</td></tr>
<tr><td rowspan="4">Faktor Perubahan Nilai Metriks</td><td>CBO</td><td>DIT</td><td>NOC</td></tr>
<tr><td>Rata-rata nilai CBO mengalami peningkatan setelah refactoring. Hal itu disebabkan oleh class activity yang mengimplementasikan data binding dikarenakan ui / xml terikat pada activity sehingga coupling pada activity meningkat. Namun penggunaan databinding juga berperan dalam meringkas jumlah code, karena tidak perlu inisialisasi layout terlebih dahulu sebagai contoh pada gambar 15 & 16 Selain itu, masih terdapat komponen-komponen yang berhubungan langsung seperti Progress Dialog, Method Helper, dan Session Manager.</td><td>Rara-rata nilai DIT mengalami peningkatan setelah refactoring. Hal itu disebabkan oleh class viewmodel mengextends library viewmodel sehingga memiliki kedalaman class yang berjumlah 2. DIT pada class activity memiliki kedalaman 9 karena mengextends library yang ada pada gambar 14.</td><td>Rata-rata nilai NOC mengalami penurunan setelah refactoring. Hal itu disebabkan oleh tidak adanya class yang mewariskan methodnya kepada class lainnya. Berbeda sebelum refactoring dimana terdapat class interface yang mewariskan methodnya untuk class activity/view.</td></tr>
<tr><td colspan="2">WMC</td><td>LCOM</td></tr>
<tr><td colspan="2">Rata-rata nilai WMC mengalami peningkatan setelah refactoring. Pada modul tambah bimbingan, class activity dan repository mempengaruhi bertambahnya rata-rata nilai WMC, dikarenakan jumlah method yang lebih banyak dan juga seringnya penggunaan if statement.</td><td>Rata-rata nilai LCOM mengalami penigkatan setelah refactoring.Hal itu dikarenakan pada modul tambah bimbingan, nilai LCOM pada class viewmodel mengalami peningkatan yang cukup drastis.pada modul tambah sidang class activity dan viewmodel.</td></tr>
</table>

Sebagai tambahan, pada penelitian ini juga tidak menerapkan prinsip MVVM yaitu untuk satu view hanya mempunyai satu viewmodel saja [15]. Contohnya pada tabel 6 terdapat class DosenSidangTambahActivity yang terhubung pada dua class viewmodel yaitu SidangViewModel & ProyekAkhirViewModel yang seharusnya hanya terhubung pada satu viewmodel. Sehingga mempengaruhi naiknya rata-rata nilai CBO.

Selanjutnya dilakukan juga analisa perbandingan sebelum dan sesudah refactoring untuk modul tambah bim- bingan S1 dan tambah sidang S1 pada gambar 12 dan 13. Hasilnya sama seperti modul tambah bimbingan D3 dan tambah sidang D3 sebelumnya. rata-rata dari nilai CBO, DIT, WMC, dan LCOM mengalami kenaikan dan NOCmengalami penurunan setelah refactoring kedalam arsitektur MVVM pada kedua modul tersebut. Sehingga pada penelitian ini arsitektur MVVM mempengaruhi penurunuan kualitas dalam aspek *reusability* karena meningkatkan *coupling*, meningkatkan kompleksitas, menurunkan kohesi pada *class*.

V. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, kesimpulan yang didapatkan yaitu:

1. *Refactoring* dapat mempengaruhi kualitas internal seperti *complexity, coupling, dan inheritance* serta kuali- tas eksternal seperti *maintainability, reusability, understandability*, dan *efficiency*. Oleh karena itu, *refacto- ring* dapat meningkatkan *software reuse* atau penggunaan kembali suatu perangkat lunak.
2. Implementasi aplikasi baseline (MVP) ke MVVM dilakukan dengan mengikuti struktur dimana activi- ty/fragment yang berperan sebagai *View* terhubung pada *class* viewmodel

yang berperan untuk menyimpan data berupa liveData agar view dapat melakukan observasi terhadap ViewModel. Dalam MVVM, tidak ada- nya *class interface* yang mewariskan *method* kepada view seperti MVP, ViewModel juga terhubung dengan repository yang bertanggung jawab sebagai model untuk mendapatkan data, dan melakukan perubahan data.Repository juga bertanggung jawab untuk mengatur sumber data yang dibutuhkan. oleh aplikasi baik itu secara lokal maupun *remote* atau web service.

3. Identifikasi *class* atau komponen yang digunakan kembali dilakukan dengan memilih *class* yang berhubung-an dengan fitur-fitur yang dijelaskan sebelumnya pada studi literature. Hasilnya terdapat 78 dari 199 *class* yang digunakan kembali.

4. Perubahan alur proses bisnis aplikasi sidang tugas akhir D3 Informatika menjadi S1 dilakukan dengan meng-ikuti requirement yang dibutuhkan dalam membangun aplikasi sidang tugas akhir S1. Dalam penelitian hanya beberapa modul yang diambil dalam membangun aplikasi sidang tugas akhir S1. Alur proses bis- nis untuk modul bimbingan tidak mengalami perubahan, namun terdapat perbedaan dari sisi fungsionalitas.Pada aplikasi baseline (D3 Informatika), mahasiswa hanya melakukan bimbingan dengan satu dosen pem- bimbing sedangkan pada aplikasi S1 Informatika, mahasiswa dapat melakukan bimbingan dengan 1 atau 2dosen pembimbing.

5. Berdasarkan hasil perbandingan pengukuran aplikasi sebelum dan sesudah *refactoring* menggunakan ar- sitektur MVVM, implementasi arsitektur MVVM memiliki rata-rata nilai CBO yang meningkat sebesar 2.25%, WMC meningkat sebesar 2.53%, LCOM meningkat sebesar 0.13%, serta menurunnya nilai NOC menjadi 0. Faktor utama kesalahan pada implementasi MVVM pada penelitian ini adalah tidak menerapk- an prinsip MVVM yaitu untuk satu view hanya mempunyai satu viewmodel saja sehingga mempengaruhi naiknya rata-rata nilai CBO. Maka dari itu, dapat dinyatakan pada studi kasus ini arsitektur MVVM mem- pengaruhi penurunuan kualitas dalam aspek *reusability* karena meningkatkan *coupling*, meningkatkan kom- pleksitas, dan menurunkan kohesi pada *class*.

6. Untuk penelitian selanjutnya, implementasi arsitektur MVVM sebaiknya digabungkan dengan metode lain yang dapat mengatasi *coupling* pada komponen-komponen yang digunakan pada suatu class, semisal dengan implementasi *design pattern* yang tepat untuk dapat meningkatkan *reusability*.

REFERENSI

[1] Measure, visualise, and improve code quality: Better code better quality!, Mar 2021.

[2] N. Akhtar and S. Ghafoor. Analysis of architectural patterns for android development.

[3] M. Anasuodei and N. A. Ojekudo. Software reusability: Approaches and challenges.

[4] S. R. Chidamber and C. F. Kemerer. Towards a metrics suite for object oriented design. In *Conference proceedings on Object-oriented programming systems, languages, and applications*, pages 197–211, 1991.

[5] A. M. Draz, M. S. Farhan, and M. M. Eldefrawi. A survey of refactoring impact on code quality. 2021.

[6] W. B. Frakes and K. Kang. Software reuse research: Status and future. *IEEE transactions on Software Engineering*, 31(7):529–536, 2005.

[7] B. M. Goel and P. K. Bhatia. Analysis of reusability of object-oriented system using ck metrics. *International Journal of Computer Applications*, 60(10):32–36, 2012.

[8] H. Källström. Increasing maintainability for android applications: Implementation and evaluation of three software architectures on the android framework, 2016.

[9] J. Kouraklis. Mvvm as design pattern. In *MVVM in Delphi*, pages 1–12. Springer, 2016.

[10] U. L. Kulkarni, Y. Kalshetty, and V. G. Arde. Validation of ck metrics for object oriented design measurement. In *2010 3rd international conference on emerging trends in engineering and technology*, pages 646–651. IEEE, 2010.

[11] X. Li, D. Chang, H. Pen, X. Zhang, Y. Liu, and Y. Yao. Application of mvvm design pattern in mes. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems(CYBER)*, pages 1374–1378. IEEE, 2015.

[12] T. Lou et al. A comparison of android native app architecture mvc, mvp and mvvm. *Eindhoven University of Technology*, 2016.

[13] J. Mago and P. Kaur. Analysis of quality of the design of the object oriented software using fuzzy logic. In *International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings published in International Journal of Computer Applications®(IJCA)*, pages 21–25. Citeseer, 2012.

[14] H. Manoj and A. Nandakumar. Constructing

relationship between software metrics and code reusability in object oriented design. *APTIKOM Journal on Computer Science and Information Technologies*, 1(2):63–76,2016.

[15] M. O. Ozturk. *Evaluating Maintainability of Android Applications: Mooncascade Case Study*, page 1–67, May 2021.

[16] N. Padhy, R. Singh, and S. C. Satapathy. Software reusability metrics estimation: algorithms, models and optimization techniques. *Computers & Electrical Engineering*, 69:653–668, 2018.

[17] L. Prayogo. *Implmentasi Pola Arsitektur Model-view-view Model Untuk Reusability Perangkat Lunak Pada Proses Sidang Akhir Fakultas Informatika Universitas Telkom*, Aug 2021.

[18] F. E. Shahbudin and F.-F. Chua. Design patterns for developing high efficiency mobile application. *Journal of Information Technology & Software Engineering*, 3(3):1, 2013.

[19] S. Singh, S. Singh, and G. Singh. Reusability of the software. *International journal of computer applications*,7(14):38–41, 2010.

[20] W. Sun, H. Chen, and W. Yu. The exploration and practice of mvvm pattern on android platform. In *2016 4th International Conference on Machinery, Materials and Information Technology Applications*. Atlantis Press,2017.

[21] A. Syromiatnikov and D. Weyns. A journey through the land of model-view-design patterns. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 21–30. IEEE, 2014.

[22] R. Van Ommering. Software reuse in product populations. *IEEE Transactions on Software Engineering*, 31(7):537–550, 2005.

[23] I. W. G. S. Wijaya and W. G. Suma. Penerapan web service pada aplikasi sistem akademik pada platform sistem operasi mobile android. *Teknik Informatika, STIKOM PGRI Banyuwangi*, 2012.

[24] B. Wisnuadhi, G. Munawar, and U. Wahyu. Performance comparison of native android application on mvpand mvvm. In *International Seminar of Science and Applied Technology (ISSAT 2020)*, pages 276–282. Atlantis Press, 2020.