Implementasi Pola Arsitektur *Model-view-view Model* Untuk *Reusability* Perangkat Lunak Pada Proses Sidang Akhir Fakultas Informatika Universitas Telkom

Tugas Akhir
diajukan untuk memenuhi salah satu syarat
memperoleh gelar sarjana
dari Program Studi S1 Informatika
Fakultas Informatika
Universitas Telkom

1301198518
Lintang Prayogo

Program Studi Sarjana S1 Informatika
Fakultas Informatika
Universitas Telkom
Bandung
<2022>

LEMBAR PENGESAHAN

Implementasi Pola Arsitektur Model-view-view Model Untuk Reusability Perangkat Lunak Pada Proses Sidang Akhir Fakultas Informatika Universitas Telkom

Implementation of the Model-View-View Model Architecture Pattern for Software reuse in the Final Session Process of the Faculty of Informatics, Telkom University

NIM : 11301198518
Lintang Prayogo

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh gelar pada Program Studi Sarjana S1 Informatika
Fakultas Informatika
Universitas Telkom

Bandung, <26/08/2021>
Menyetujui

Pembimbing I,	Pembimbing II,
Yudi Priyadi, S.T, M.T	Shinta Yulia Puspitasari, S.T, M.T
NIP: 20710004	NIP: 13880046

Ketua Program Studi
Sarjana S1 INFORMATIKA,

Dr. Erwin Budi Setiawan, S.Si., M.T.
NIP: 00760045

LEMBAR PERNYATAAN

Dengan ini saya, Lintang Prayogo, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul <Judul TA> beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya,

Bandung, 14 Februari 2021
Yang Menyatakan

Lintang Prayogo

Implementasi Pola Arsitektur *Model-view-view Model* Untuk *Reusability* Perangkat Lunak Pada Proses Sidang Akhir Fakultas Informatika Universitas Telkom

Lintang Prayogo[1],Dawam Dwi Jatmiko Suwawi, S.T., M.T.[2], Shinta Yulia Puspitasari, S.T, M.T.[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung [1]prayogolintang@students.telkomuniversity.ac.id, [2]whyphi@telkomuniversity.ac.id,[3]shintayulia@telkomuniversity.ac.id

Abstrak

Dalam proses pengembangan aplikasi waktu adalah sebuah faktor yang penting. Ketika kompleksitas terus meningkat tetapi waktu yang tersedia terbatas [1]**, waktu yang digunakan untuk menulis kode menghabiskan kurang lebih 40%** [2]**.Untuk mempersingkat waktu solusi umum yang digunakan adalah daur ulang aplikasi**[3]**. Daur ulang aplikasi menjadi penting untuk membagun aplikasi yang murah dan tangguh. Aplikasi hasil daur ulang juga memiliki tingkat cacat lebih rendah dari yang tidak**[4] **. Reusability adalah kemampuan asset aplikasi untuk didaur ulang[5].Pada penelitian ini meneliti dampak penerapan *model-view viewmodel (MVVM), dependency injection dan template method* terhadap aspek CK metrics yang berkaitan pada aspek reusability. terdapat 5 aspek yang berkaitan dengan *reusability* yaitu Weight Method Class (WMC), Depth of Inheritance Tree (DIT) , Number of Children(NOC),Coupling Between Object Classes(CBO),Lack of Cohesion of Methods (LCOM)[8][9].Hasil penelitian menunjukan CBO,LCOM,WMC,DIT aspek ini mengalami peningkatan kualitas yang terlihat jelas.Pada aspek NOC ada peningkatan tetapi hanya 0,1 dari sebelum proses daur ulang.Dapat disimpulkan penerapan 3 metode tersebut berdampak baik pada reusability . 4 dari 5 atribut *CK metrics* berkaitan dengan reusability mengalami perubahan yang terlihat . Selain itu juga kombinasi pola desain atau metode tertentu mungkin menjadi kunci dalam peningkatan kualitas perangkat lunak**

Keywords: MVVM,*Dependency Injection,Template Method,Reusability*

1. Pendahuluan

Latar Belakang

Dalam proses pengembangan aplikasi waktu adalah sebuah faktor yang penting. Ketika kompleksitas terus meningkat tetapi waktu yang tersedia terbatas [1], waktu yang digunakan untuk menulis kode menghabiskan kurang lebih 40% [2].Untuk mempersingkat waktu solusi umum yang digunakan adalah daur ulang aplikasi[3]. Daur ulang aplikasi menjadi penting untuk membagun aplikasi yang murah dan tangguh. Aplikasi hasil daur ulang juga memiliki tingkat cacat lebih rendah dari yang tidak[4] .

Reusability adalah kemampuan asset aplikasi untuk didaur ulang[5]. Untuk menentukan kualitas reusability akan diukur dengan CK Metrics. CK metrics merupakan ukuran kualitas untuk pemrograman berbasis objek(PBO) dikarenakan kotlin atau java merupakan PBO dalam aplikasi android native [6][7]. Pada Metrics ini terdapat 5 aspek yang berkaitan dengan reusability yaitu Weight Method Class (WMC), Depth of Inheritance Tree (DIT),Number of Children(NOC),Coupling Between Object Classes(CBO),Lack of Cohesion of Methods (LCOM)[8][9].Untuk memperbaiki kondisi atribut untuk mendukung reusability.Yaitu model-view viewmodel (MVVM), dependency injection dan template method.

MVVM memiliki beberapa atribut yang mungkin meningkatkan reusability. Data binding merupakan salah atribut yang dimiliki MVVM yang mengurai coupling antara view dan model hal ini akan berdampak juga terhadap kualitas CBO yang lebih rendah dan baik untuk reusability[10]. Viewmodel pada MVVM memiliki pemisahan fokus yang lebih baik. Sehingga berdampak positif pada tingkat coupling dan cohesion[11][12][13].Hal ini berkaitan dengan CBO dan LCOM.Semakin rendah tingkat CBO dan LCOM maka akan semakin baik juga tingkat reusability[8][9]. MVVM juga adalah pola arsitektur yang didukung oleh google untuk aplikasi android native dan [14].

Dependency Injection (DI) telah diadopsi sebagai pola desain untuk menangani kompleksitas pada aplikasi. Aplikasi yang tidak menerapkan konsep Dependency yang berdampak pada kesulitan perawatan dan duplikasi kode pada aplikasi dan tingkat coupling yang tinggi[15]. DI juga dapat menurunkan CBO dan LCOM sebesar 10%[16].

Template method adalah salah satu pola desain GoF yang menggunakan prinsip pewarisan[17]. Pola ini sangat untuk diterapkan pada kelas identik.Template method dapat meningkatkan nilai NOC dan DIT[18]. Peningkatan NOC dan DIT baik untuk reusability[8][9]. Template method , Adapter-Command, Singleton dan State-Strategy lebih unggul dalam aspek reusability dibanding pola GoF lainya[19].

Terakhir aspek WMC mewakili kompleksitas kode dalam suatu kelas. Untuk menurunkan tingkat WMC dalam suatu kelas untuk reusability yang lebih baik[8][9]. DI mengurangi kompleksitas dengan mengurangi duplikasi kode yang ada[15]. View model memiliki pemisahan fokus yang lebih jelas sehingga kode tidak terlalu kompleks[13]. Terakhir template dapat mengurangi kompleksitas dikarenakan kelas abstrak sudah mendefinisikan beberapa langka-langkah dalam suatu metode[17][18].

Berdasarkan paparan diatas di atas, penelitian ingin turut serta dalam mengimplementasikan 3 metode untuk mengetahui dampak dari penggunaan metode terhadap reusability. Untuk itu kami mengetahui hal tersebut kami memilih aplikasi sidang tugas akhir D3 Informatika untuk kami daur ulang menjadi aplikasi S1 Informatika. Selain itu juga diharapkan penelitian ini dapat menghasilkan aplikasi baru yang dapat di daur ulang.

Rumusan Masalah

Berdasarkan latar belakang tersebut didapatkan rumusan masalah yang akan ditinjau dalam penelitian ini, yaitu:

1. Bagaimana elisitasi untuk mendapatkan kebutuhan dalam proses implementasi dan proses modifikasi atau daur ulang aplikasi D3 Informatika menjadi aplikasi S1 Informatika yang akan dilaksanakan ?.
2. Bagaimana metode dalam pembuatan model bisnis sidang S1 Informatika menggunakan model UML ?
3. Bagaimana melakukan identifikasi aplikasi D3 Informatika ?
4. Bagaimana melakukan proses modifikasi atau daur ulang kode pada aplikasi proyek akhir (PA) D3 Informatika menjadi aplikasi S1 Informatika ?
5. Bagaimana mengukur atribut *ck metrics* terkait reusability ?.
6. Bagaimana dampak penerapan 3 metode ini terhadap atribut ck metrics terkait *reusability ?*

Tujuan

Tugas akhir ini secara garis besar memiliki tujuan dalam implementasi konsep MVVM, template method, dan dependency injection dan menghasilkan *base application* yang baru, sehingga terjadi peningkatan kualitas *reusability* pada perangkat lunak yang digunakan dalam proses sidang tugas akhir(TA) S1 Informatika Universitas Telkom. Untuk mencapai tujuan umum tersebut, maka pada kegiatan ini akan menerapkan tujuan khusus sebagai berikut:

1. Melakukan elisitasi untuk mendapatkan kebutuhan dalam proses implementasi dan modifikasi aplikasi D3 Informatika menjadi aplikasi S1 Informatika yang akan dilaksanakan.
2. Membuat model UML berdasarkan proses bisnis sidang TA S1 Informatika perangkat lunak agar dapat menjelaskan aktor yang terlibat dalam proses sidang akhir S1 Informatika
3. Melakukan identifikasi aplikasi D3 Informatika guna menentukan fragment code reuse berdasarkan kemiripan pada model yang telah dibuat dan mendeteksi *code smell*
4. Berdasarkan model UML,*fragment code reuse* dan *code smell* tersebut, akan dilakukan kegiatan proses daur ulang kode dan penerapan MVVM, template Method dan dependency injection.
5. Melakukan pengukuran atribut ck metrics terkait reusability.
6. Mengetahui dampak penerapan 3 metode ini terhadap *reusability*

2. Studi Terkait

Pola Desain dan Template Method

Pola desain adalah gambaran tentang suatu kelas serta setiap objek yang berinteraksi, yang bertujuan untuk menyelesaikan sebuah masalah generik dalam konteks tertentu. Pola desain merupakan solusi pada rekayasa lunak yang dapat diterapkan berulang kali. Pola desain adalah panduan yang menjabarkan bagaimana sebuah masalah diselesaikan dan dapat diterapkan kembali pada masalah yang berbeda. Pada pemrograman berbasis objek, pola desain biasanya menjelaskan interaksi antar kelas dan keterkaitan antar objek.

Merujuk pada tabel 1 dijelaskan ada berbagai macam pola desain yang dapat diterapkan sesuai dengan tujuan yang ingin dicapai dalam pembuatan suatu perangkat lunak. Beberapa jenis pola desain yang cukup populer adalah 23 macam pola desain yang dikenalkan oleh *The Gang of Four (GoF).* Secara umum, ke-23 macam pola desain tersebut dapat dikelompokkan menjadi 3 kategori berdasarkan sifatnya yaitu patterns creational, patterns behavioral dan patterns structural. Berikut ini adalah tabel yang berisi nama-nama pola desain dari Erich Gamma, Richard Helm, Ralph Johnson dan John Vlissides "*Gang of Four*" [17].

Tabel 1. Tabel Pola Desain GoF

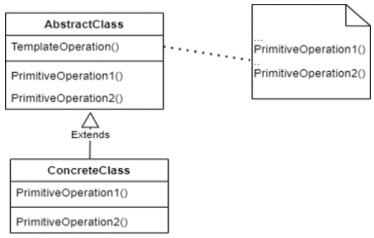
Creational	Behavioral	*Structural*
Factory Method	*Adapter*	*Chain of Responsibility*
Abstract Factory	*Bridge*	*Command*
Builder	*Composite*	*Iterator*
Prototype	*Decorator*	*Mediator*
Singleton	*Façade*	*Memento*
	Flyweight	*Observer*
	Proxy	*State*
		Strategy
		Template Method
		Visitor

Template method merupakan salah satu dari patterns behavioral yang menjabarkan pembentukan suatu kerangka algoritma, yang akan menerapkan suatu proses lebih spesifik di dalam sebuah kelas turunannya. Tujuan dari hal ini adalah menerapkan suatu standar pada pembuatan suatu kelas [17]. Sebuah pola dalam

template method harus menggunakan prinsip sebagai berikut :

1. Dalam implementasi bagian yang berbeda pada sebuah algoritma sekali atau menurunkan karakteristik pada sebuah kelas turunan.
2. Menurunkan karakteristik yang berbeda haruslah dilakukan pada kelas turunan. Ketika karakteristik umum diantara kelas turunan sebagai faktor dalam mendefinisikan kelas abstrak, untuk menangani sifat-sifat yang berbeda yang sudah didefinisikan untuk mencegah terjadinya duplikasi, maka sebaiknya identifikasi kode yang berbeda dan bagi keadaan dalam sebuah operasi baru, dan ganti kode tersebut dengan template method, lalu panggil dengan operasi baru.
3. Untuk mengontrol sebuah ekstensi kelas turunan, maka definisikan operasi pada template method yang dapat mengaitkan sebagai poin spesifik yang memberikan akses agar dapat digunakan.

Merujuk pada gambar 1 menjelaskan bagaimana suatu kelas berinteraksi. Suatu kelas abstrak akan berisi operasi template yang akan digunakan pada banyak kelas lainya dan operasi primitif. Operasi primitif adalah operasi umum yang digunakan pada kelas lain sedangkan operasi template adalah operasi yang dapat diimplementasikan pada kelas turunannya[17].



Gambar 1. Struktur Template Method

Dependency Injection

Dalam pengembangan aplikasi, Dependency Injection adalah teknik dimana suatu objek menerima objek lain yang bergantung padanya. Objek lain ini disebut dependensi. Dalam hubungan "using" yang unik, objek penerima disebut klien dan objek yang diteruskan (yaitu, "di injeksi") disebut service[14]. *Dependency Injection* juga merupakan salah satu dari prinsip inversion of control . Inversion of control memungkinkan kerangka untuk mengambil kendali atas alur eksekusi program dan mengirimkan panggilan ke kode tertulis. Manfaat menggunakan inversion of control adalah sebagai berikut :

1. Transisi yang lebih mudah antara implementasi yang berbeda,
2. Modularitas program yang lebih baik,
3. Pengetesan program yang lebih mudah dalam mengisolasi komponennya

Manfaat utama dari dependency injection adalah *loose coupling*. Suatu objek dapat di test dan dibentuk secara mandiri dikarenakan objek tidak bergantung pada objek lainya. Dengan menerapkan , pengujian jauh lebih mudah karena memungkinkan membuat objek palsu. Objek palsu adalah objek simulasi yang meniru perilaku objek nyata dengan cara yang terkontrol. Objek tiruan adalah biasanya dibuat untuk menguji perilaku beberapa objek lain[15].

Dalam mendukung penerapan dependency injection terdapat library yang kita gunakan yaitu hilt. Hilt adalah library dependency injection untuk Android yang mengurangi boilerplate ketika melakukan dependency injection manual dalam aplikasi android. Dengan melakukan dependency injection manual, Mengharuskan membuat setiap kelas dan objeknya secara manual. Hilt dapat menginjeksikan beberapa kelas yaitu application,activity,fragment,viewservice,broadcast receiver

MVVM

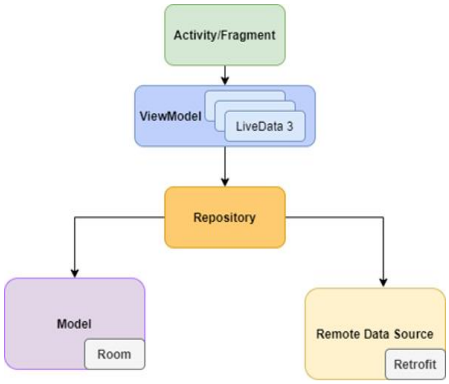
Merujuk pada gambar 2 terdapat 3 lapisan yaitu View,ViewModel dan Model. Dalam pola MVVM, ViewModel menggantikan Presenter dan Controller. Pada pola ini ViewModel dan View memiliki tanggung jawab yang berbeda[20]. View akan mengetahui perubahan yang dilakukan oleh ViewModel tetapi ViewModel

tidak mengetahui keberadaan ViewModel oleh karena itu ini juga dapat disebut kesadaran satu arah. ViewModel akan meneruskan input dan data kepada model setelah itu melacak setiap perubahan yang dibuat oleh Model lalu memberitahukan kepada View.

Gambar 2. Sumber: [20]

Android Architecture Components adalah kumpulan library yang akan menangani beberapa hal terkait masukkan dari user secara langsung, penyimpanan ke basis data lokal, dan menampilkan data yang sudah olah, sehingga dapat menghasilkan aplikasi yang tangguh, dapat diuji, dan dapat dipelihara secara berkelanjutan. Diagram berikut menunjukkan bagaimana semua modul harus berinteraksi satu sama lain setelah mendesain aplikasi.

Berdasarkan gambar 2.3 setiap komponen hanya bergantung pada komponen yang berada satu tingkat di bawahnya. Misalnya, aktivitas dan fragment hanya bergantung pada model tampilan. *Repository* adalah satu-satunya kelas yang bergantung pada beberapa kelas lainnya. Dalam contoh ini, *repository* bergantung pada model data persisten dan sumber data yang diambil dari internet[15].



Gambar 3. Diagram *Architecture Components*.Sumber: [14]

CK Metrics

CK Metrics (Chidamber-Kemerer Metrics) terdiri dari 6 metrik yang dijadikan parameter dalam mengukur kualitas yaitu Weight Method per Class, Depth of Inheritance Tree, Number of Children, Coupling Between Object Classes , Lack of Cohesion of Methods dan terdapat 4 metrik yang berpengaruh terhadap reusability adalah sebagai berikut[6][9] :

1. *Depth of Inheritance Tree* (DIT)

DIT merupakan jarak simpul terhadap *root* dalam kelas yang menerapkan konsep inheritance atau dengan kata lain berapa banyak *inheritance* yang digunakan dalam kelas tersebut. Semakin tinggi nilai DIT menunjukan desain yang lebih kompleks dan *reusability* yang lebih baik.

2. *Number of Children* (NOC)

NOC merupakan jumlah kelas turunan langsung yang terdapat dalam sebuah class. NOC mengukur berapa banyak kelas turunan yang menurunkan *method* dari kelas induk. Nilai NOC yang tinggi mengidentifikasikan *reusability* yang lebih baik.

3. Coupling Between Object Classes (CBO)

CBO menghitung *coupling non-inheritance* antara suatu kelas terhadap kelas lain, contohnya suatu kelas menggunakan *method* atau variabel dari kelas lain. Jika suatu kelas memanggil beberapa *method* maupun variabel

dari sebuah kelas lain, maka jumlah relasi tetap dihitung. Nilai CBO yang tinggi mengidentifikasikan kurangnya *reusability, efficiency, dan maintainability*.

4. Weight Method per Class (WMC).

WMC merupakan metrik yang berfokus pada kompleksitas dari *method* pada sebuah kelas. Kompleksitas dapat dihitung dengan rumus *cyclomatic complexity*. WMC menjumlahkan semua nilai kompleksitas dalam semua *method* di dalam sebuah kelas.

seperti persamaan dibawah ini.

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

dimana:

n = jumlah *method* dalam sebuah kelas

c_i = *cyclomatic complexity* dari sebuah *method i*

5. Lack of Cohesion of Methods (LCOM)

LCOM menghitung selisih antara jumlah method dalam sebuah class yang memiliki parameter atau return type. Semakin besar nilai LCOM, semakin besar pula kompleksitas class tersebut dan berpengaruh pada rendahnya tingkat reusability dan efficiency. LCOM dapat dihitung sesuai Persamaan 2 dan 3.

$$LCOM3 = (m - sum(mA)/a) / (m-1) \qquad (2)$$

m = jumlah prosedur (metode) di kelas
a=sejumlah variabel (atribut) di dalam kelas. a berisi semua variabel baik yang dibagikan (statis) atau tidak.
mA=sejumlah metode yang mengakses variabel (atribut)
sum(mA) =jumlah mA atas atribut suatu kelas

Penelitian Terkait

Berdasarkan Tabel 2 dapat dicermati posisi kegiatan ini, yang melakukan studi literatur untuk mendapatkan rujukan yang dikutip dalam Tugas Akhir ini. Silakan dicermati pada studi literatur ke 1-9.

Tabel 2. Tabel Penelitian Terkait

No.	Identitas Penelitian	Pembahasan	Metode	Hasil Penelitian
1	H. Y. Yang, E. Tempero, and H. Melton, "An empirical study into use of dependency injection in Java," Proc. Aust. Softw. Eng. Conf. ASWEC, pp. 239–247, 2008, doi: 10.1109/ASWEC.2008.4483212.	Dalam studi ini, kami membahas penggunaan "dependency injection",yang merupakan metode desain yang diklaim dapat meningkatkan atribut kualitas desain aplikasi seperti ekstensibility, modifiability, testability, dan reusability. Penulis mengembangkan definisi operasional untuk itu dan teknik analisis untuk mendeteksi penggunaannya. Dengan mengujinya pada 34 aplikasi terbuka.	1. Dependency injection 2. Constructor no default (CND) 3. Method no default (MND) 4. Constructor with default (CWD) 5. Method with default (MWD)	Hasil CND , MND,MWD,CWD,MWD

2	S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, 1994, doi: 10.1109/32.29589.	Membahas pembuatan metrik untuk mengukur kualitas aplikasi pada PBO.terdiri dari 6 atribut sebagai berikut : 1. *Weight Method Class (WMC),* 2. *Depth of Inheritance Tree (DIT)* 3. *Number of Children(NOC)* 4. *Coupling Between Object Classes(CBO)* 5. *Lack of Cohesion of Meth-ods(LCOM* 6. *Response For Class(RFC)*	1. PBO 2. Statistik	*1. CK metrics* *2. Histogram*
3	M. H. M. and N. A. N., "Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design," *APTIKOM J. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 63–76, 2016, doi: 10.34306/csit.v1i2.49	Pada karya tulis ini mencari tahu relasi antara *reusability* dan *ck metrics.* atribut yang terkait adalah sebagai berikut: 1. Weight Method Class (WMC) 2. Depth of Inheritance Tree (DIT) 3. Number of Children(NOC),Coupling Between Object Classes(CBO) 4. Lack of Cohesion of Meth-ods(LCOM	*1. CK metrics* 2. *Stochastic Assessment Model*	Grafik relasi aspek ck metrics pada *reusability*
4	P. Mago, J. & Kaur, "Analysis of quality of the design of the object oriented software using fuzzy logic," Int. Conf. Recent Adv. Futur. Trends Inf. Technol. Proc. Publ. Int. J. Comput. Appl., pp. 21–25, 2012, [Online]. Available: https://pdfs.semanticscholar.org/043f/ddd25c3af905a24d762ee832221170f4bcb7.pdf.	Membahas pembentukan program untuk mengukur reusability dengan ck metrics menggunakan *fuzzy algorithm* .Keterkaitan dengan karya tulis ini karena sama-sama membahas aspek-aspek pada *ck metrics* terhadap *reusability*	*1. CK metrics* *2. Algorithm*	1. Alat ukur reusability dengan CK metrics

5	X. Li, D. Chang, H. Pen, X. Zhang, Y. Liu, and Y. Yao, "Application of MVVM design pattern in MES," 2015 IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. IEEE-CYBER 2015, no. 2012, pp. 1374–1378, 2015, doi: 10.1109/CYBER.2015.7288144.	Membahas masalah pada pola arsitektur MVC dan menjelaskan tahapan-tahapan bagaimana seorang programmer dapat melakukan implementasi ke pola arsitektur MVVM.Penulis beranggapan MVC memiliki kualitas *maintainability* yang lebih buruk daripada MVVM. Penulis menjelaskan bahwa objek ViewModel dapat membagi fokus masalah pada kode dengan lebih baik sehingga kompleksitas kode dan hal itu juga menyebabkan penggunaan kembali kode lebih baik	1. MVVM *2. .Net Framework*	1. Aplikasi manufaktur 2. Desain lapisan model 3. Desain lapisan view 4. Desain lapisan viewmodel 5. Desain lapisan
7	F. Sholichin, M. A. Bin Isa, S. A. Halim, and M. F. Bin Harun, "Review of ios architectural pattern for testability, modifiability, and performance quality," J. Theor. Appl. Inf. Technol., vol. 97, no. 15, pp. 4021–4035, 2019.	Membahas kemampuan 4 pola arsitektur . Untuk tingkat kemudahan modifikasi diwakilkan dengan *cohesion* dan *coupling* level. Kaitan dengan penelitian pola arsitek mvvm memiliki coupling level terbaik setelah viper dan cohesion level yang terbaik.Hal ini menjadi landasan pemilihan pola MVVM	1. MVVM 2. MVC 3. MVP 4. VIPER *5. Cohesion Level* 6. *Coupling Level*	1. Tabel *modifiability* 2. Tabel *testability*
8	Syromiatnikov, A., & Weyns, D. (2014). A Journey through the Land of Model-View-Design Patterns. 2014 IEEE/IFIP Conference on Software Architecture. doi:10.1109/wicsa.2014.13	Studi ini bertujuan untuk lebih memperjelas ragamnya pola desain MV * dan membantu praktisi menjadi lebih baik dalam pemilihan pola. MVC pola desain ini merupakan nenek moyang dari 2 pola lainya dan sangat cocok untuk pembuatan website. Namun pola desain memiliki keterikatan sangat tinggi pada controller dan view. MVVM pola desain ini mendukung satu sumber data untuk beraneka ragam tampilan. Namun MVVM memiliki performa yang lebih rendah dikarenakan proses sinkronisasi	1. MVVM 2. MVP 3. MVC	1. Tabel pro kontra MVVM,MVC,MVP 2. Aplikasi pengisian formulir kampus

9	M. T. H. E. State and L. S. Technology, "Usage of Dependency Injection within different frameworks," no. March, 2020.	Membahas penerapan DI dengan bantuan *spring framework.* Dalam memberikan contoh DI bagian front-end dan back-end .Penerapan di membuat tingkat coupling lebih rendah	1. Spring framework. 2. DI 3. MVC	1. Contoh kode
10	E. Razina and D. Janzen, "Effects of dependency injection on maintainability," Proc.	Karya tulis ini membahas dampak penggunaan DI .Pada aspek maintabilitas aplikasi. DI menunjukan indikasi mendukung low coupling dan high cohesion. Indikasi berupa penurunan tingkat LCOM dan CBO sebesar 10%	1. CK Metrics 2. DI	Tabel Perbandingan LCOM,CBO,RFC
11	Rochimah, S., Nuswantara, P. G., & Akbar, R. J. (2018). Analyzing the Effect of Design Patterns on Software Maintainability: A Case Study. 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS). doi:10	Pada penelitian yang dilakukan penulis sebelumnya penerapan pola desain pada aplikasi kemahasiswaan tidak menunjukkan hasil yang baik pada aspek perawatan. Kali ini penulis menggunakan domain logic sebagai pola desain. Pengujian dilakukan dengan menggunakan 25010 dan ISO 25023.	1. Domain Logic 2. ISO 25010 3. ISO 25023	1. Tabel hasil pengamatan 2. *Pattern-based Design*
12	I. Binanto, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman, and B. Soewito, "Measuring the quality of various version an object-oriented software utilizing CK metrics," 2018 Int. Conf. Inf. Commun. Technol. ICOIACT 2018, vol. 2018-January, pp. 41–44, 2018, doi: 10.1109/ICOIACT.2018.8350760.	Tujuan dari penelitian ini adalah untuk mengevaluasi Statcato untuk mengetahui kualitas aplikasi ini selama siklus hidupnya. Itu nilai metrik yang dihitung untuk berbagai rilis digunakan sebagai dasar untuk mengevaluasi kualitas aplikasiStatcato. Alat bantu yang digunakan adalah spinellis sebagai alat untuk mengukur nilai CK metrics. Statcato adalah proyek terbuka dan aplikasi berbasis java tersedia pada sistem operasi windows dan mac os	1. CK metrics	Hasil tes kelayakan pada aspek testing

13	Cherdsakulwong, N., & Suwannasart, T. (2019). Impact Analysis of Test Cases for Changing Inputs or Outputs of Functional Requirements. Proceedings - 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019, 179–183. https://doi.org/10.1109/SNPD.2019.8935754	Penulis memperkenalkan Xdroid untuk memenuhi dependencies pada saat melakukan testing. Alat bantu yang sudah ada tidak memiliki cakupan yang luas dikarenakan keterikatan pada logika bisnis	1. Method Hooking 2. Dependency Injection 3. Xposed Framework	1. Xdroid 2. Grafik xdroid 3. Hasil Uji Xdroid
14	Hussain, S., Keung, J., & Khan, A. A. (2017). The Effect of Gang-of-Four Design Patterns Usage on Design Quality Attributes. 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). doi:10.1109/qrs.2017.37	Pada penelitian ini berfokus tentang menganalisis tentang 3 yaitu hal sebagai berikut : 1. korelasi antara penggunaan pola desain dan kualitas desain atribut 2. efek perancu dari ukuran sistem (jumlah kelas) desain yang digunakan mempengaruhi kualitas desain dirilis berikutnya dari suatu sistem	1. Qmood model 2. Pola desain GoF 3. Ck Metrics	1. Hasil tes Friedman 2. Grafik pola desain terhadap atribut kualitas
15	F. Khomh and Y. G. Guéhéneuc, “Do design patterns impact software quality positively?,” Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR, pp. 274–278, 2008, doi: 10.1109/CSMR.2008.4493325	Mencari penjelasan apakah design patterns memberikan dampak yang selalu baik terhadap kualitas aplikasi dengan melakukan survey serta studi literatur. Walaupun beberapa penelitian sebelumnya menunjukan hasil positif namun beberapa hasil penerapan pola desain menghasilkan smell-code	Studi literatur dan survei	Tabel korelasi pola desain dengan komponen aplikasi

16	Roubtsov, S., Serebrenik, A., & van den Brand, M. (2010). Detecting Modularity “Smells” in Dependencies Injected with Java Annotations. 2010 14th European Conference on Software Maintenance and Reengineering. doi:10.1109/csmr.2010.45	Fleksibilitas yang ditawarkan oleh dependency injection terkadang harus ditukarkan dengan kesulitan dalam perawatan. Dalam menangani hal tersebut metode yang dapat adalah java annotations	1. Java annotations 2. Dependency injection	Tabel kesalahan penerapan anotasi pada dependency injection
17	Syromiatnikov, A., & Weyns, D. (2014). A Journey through the Land of Model-View-Design Patterns. 2014 IEEE/IFIP Conference on Software Architecture. doi:10.1109/wicsa.2014.13	Dalam aplikasi dibutuhkan antarmuka untuk pengguna .Tujuan penelitian ini mencari pro dan kontra pada setiap pola arsitektur. Penelitian ini menunjukkan pemilihan pola harus mempertimbangkan kasus penggunaan dan persyaratan kualitas , dan teknologi yang akan digunakan	1. MVVM 2. MVP 3. MVC Pola desain GoF	1. Aplikasi windows dengan penerapan MVC 2. Aplikasi windows dengan penerapan MVVM 3. Aplikasi windows dengan penerapan MVP 4. Tabel pro dan kontra pola arsitektur

18	Ampatzoglou, A., Chatzigeorgiou, A., Charalampidou, S., & Avgeriou, P. (2015). The Effect of GoF Design Patterns on Stability: A Case Study. IEEE Transactions on Software Engineering, 41(8), 781–802. doi:10.1109/tse.2015.2414917	Pada karya tulis ini dibahas bagaimana efek dari pola desain terhadap stabilitas suatu perangkat lunak. Tujuan dari studi kasus ini adalah untuk mengetahui stabilitas kelas yang terlibat dalam pola desain GoF. Untuk mencapai tujuan ini, penulis membandingkan stabilitas kelas yang berpartisipasi dalam nol, satu, atau lebih kejadian pola desain, melalui studi terhadap banyak kasus	1. Pola GoF 2. Ripple effects measurement (REM)	1. Aplikasi windows dengan penerapan MVC 2. Aplikasi windows dengan penerapan MVVM 3. Aplikasi windows dengan penerapan MVP 4. Tabel pro dan kontra pola
20	Lim, Y., Kim, M., Jeong, S., & Jeong, A. (2008). A Reuse-Based Software Development Method. 2008 International Conference on Convergence and Hybrid Information Technology. doi:10.1109/ichit.2008.190	Penelitian ini membahas kunci dalam penggunaan kembali kode. Untuk memperoleh hal ini penulis berhipotesa dengan menerapkan prinsip *low coupling.* Metode untuk memperoleh *low coupling* adalah sebagai berikut: • *Active Binding Technology* • *Active Binding Component Structure and Implementation* • *Active Binding Automation Tool* • *Dependency Injection*	1. Active Binding Technology 2. Active Binding Component Structure and Implementation 3. Active Binding Automation Tool 4. Dependency Injection	1. Mean replicati-on 2. Number of treatment and control 3. Mean score metrics
22	Jiang, S., & Mu, H. (2011). *Design patterns in object oriented analysis and design. 2011 IEEE 2nd International Conference on Software Engineering and*	Mempelajari pola desain sebagai solusi daur ulang kode. Dengan menerapkan pola desain membantu pengembang berkomunikasi dengan arsitektur perangkat lunak	1. Iterator pattern 2. Strategy pattern 3. Adapter pattern	1. Prinsip desain

	Service Science. doi:10.1109/icsess			
23	Pradhan, P., Dwivedi, A. K., & Rath, S. K. (2015). Impact of Design Patterns on Quantitative Assessment of Quality Parameters. 2015 Second International Conference on Advances in Computing and Communication Engineering. doi:10.1109/icacce.2015.102	Tujuan dari penelitian ini adalah menyediakan metodologi yang dapat membandingkan kualitas atribut pada aplikasi dengan penerapan pola dan yang tidak. Dengan menggunakan pendekatan ini, metrik yang diorientasikan secara objektif dihitung berdasarkan jumlah kelas dan relasinya dalam UML	1. QMOOD	1. Formula skor untuk penerapan pola desain 2. Formula skor untuk tanpa penerapan pola desain
24	Mikkonen, Tommi & Taivalsaari, Antero. (2019). Software Reuse in the Era of Opportunistic Design. IEEE Software. 36. 105-111. 10.1109/MS.2018.2884883.	Membahas tentang daur ulang di era yang oportunis.Pembahasan ini meliputi evolusi perangkat lunak,arsitektur perangkat lunak, motivasi serta pemilihan komponen	1. Survey 2. Review literature	Presentase daur ulang kode pada proyek terbuka
24	Higo, Y., Ohtani, A., Hayashi, S., Hata, H., & Shinji, K. (2015). Toward Reusing Code Changes. 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. doi:10.1109/msr.2015.43	Dalam jurnal ini mengajukan metode novel sistem untuk mempermudah pengembang dalam memodifikasi kode yang sudah ada.Tidak seperti teknik yang ada, sistem ini dapat memprediksi perubahan kode berdasarkan perubahan sebelumnya yang telah tersimpan pada basis data	1. Change Pattern Database 2. Change Pattern Search Engine 3. Code Search Engine	1. Grafik rasio perubahan kode 2. Tabel deteksi & perubahan kode
PERBANDINGAN LITERATUR TERHADAP SKRIPSI INI				
25	Lintang Prayogo (penelitian ini)	Membahas dampak penerapan *template method, dependency injection*,MVVM pada aplikasi berbasis android terhadap aspek *reusability*	1. Template Method 2. CK Metrics 3. *Dependency injection*	1. Aplikasi Tugas Akhir 2. Hasil Uji Pada

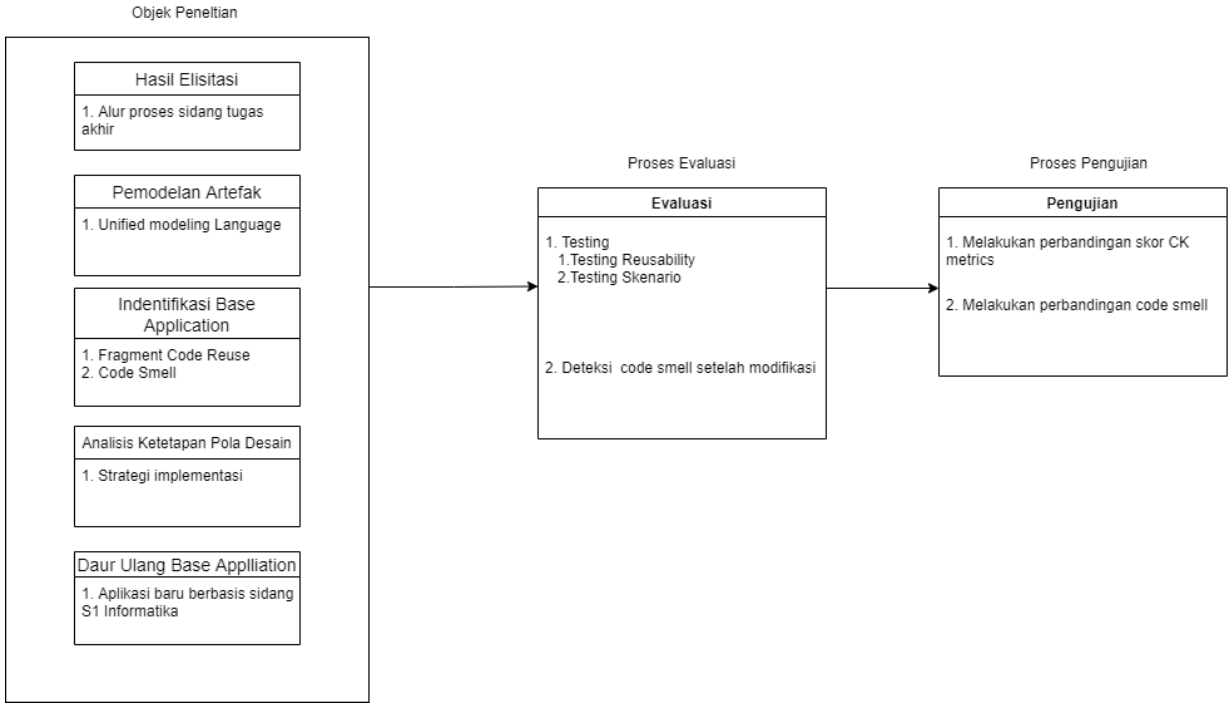
			4. MVVM	Aspek reusability

3. Metodologi

Blok Diagram

Merujuk Gambar 4, disajikan mengenai Blok Diagram Perancangan Sistem. Di dalam Blok Diagram tersebut terdiri dari tiga kegiatan utama yang dilakukan, yaitu:

1. Analisis Objek Penelitian, dibagi menjadi 3 proses, sebagai berikut :
   a) Hasil Elisitasi . Pada Hasil Elisitasi terdapat 2 metode yang digunakan, yaitu melakukan observasi untuk mendapatkan data langsung berdasarkan buku panduan tugas akhir S1 Informatika yang sedang berlaku, lalu Analisis Kebutuhan untuk menentukan kebutuhan yang tepat berdasarkan cakupan proses bisnis yang akan diakomodir aplikasi.
   b) Pemodelan Artefak. Pada tahap ini dilakukan pemodelan berdasarkan analisis kebutuhan dari proses hasil elisitasi . Pemodelan yang dilakukan yaitu Unified Modelling Language (UML).
   c) Identifikasi Base Application.Identifikasi Base Application yaitu dilakukan dengan menentukan fragment code reuse fitur pada base application dan mendeteksi code smell menggunakan CODE MR.
   d) Daur Ulang base application
2. Proses analisis ketetapan pola desain. Pada proses ini menentukan bagian apa saja yang akan diterapkan metode MVVM,*template method* dan DI. Hal ini dilakukan agar *code smell* tidak terbawa pada aplikasi modifikasi dan mendapatkan skor yang lebih untuk *ck metrics*
3. Proses Evaluasi. Evaluasi yang akan dilakukan yaitu testing dengan metode CK metrics untuk mengukur reusability dan mendeteksi code smell setelah modifikasi. Kedua nya dilakukan dengan menggunakan CODE MR.Selain itu juga melakukan testing skenario untuk memastikan proses bisnis pada aplikasi sudah benar
4. Proses validasi .Proses validasi dilakukan dengan membandingkan kondisi skor ck metrics pada base application dan aplikasi modifikasi sebagai contoh jika atribut CBO,LCOM,WMC mengalami penurunan berarti baik untuk *reusability* dan penurunan atribut DIT serta NOC mengalami peningkatan berarti baik untuk *reusability*. Selain juga membandingkan kondisi code smell pasca modifikasi hal ini dilakukan agar base application yang baru tidak memiliki masalah yang sama.

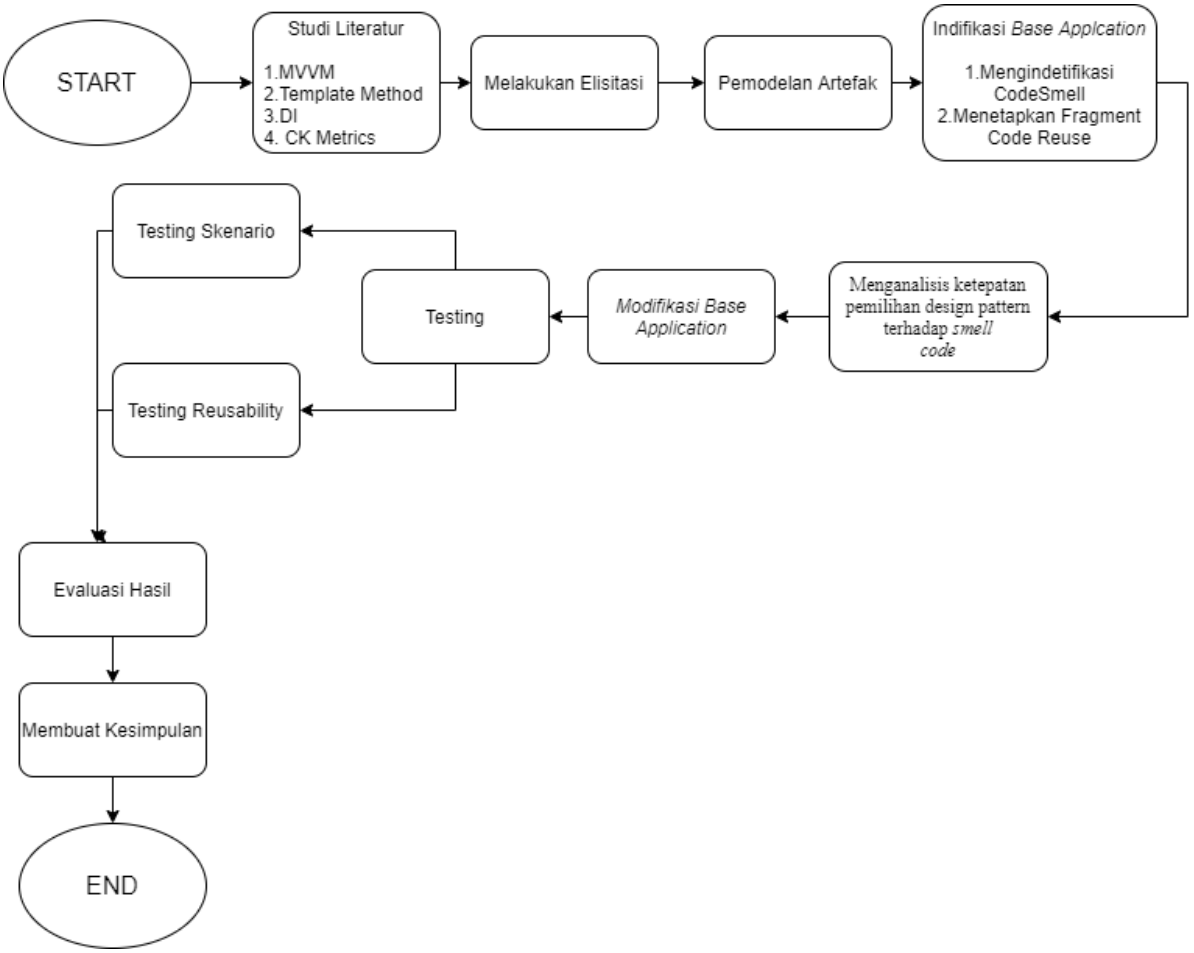


Gambar 4. Blok Diagram

Alur Pemodelan Kegiatan

Merujuk Gambar 3.2, terdapat alur kegiatan yang dilakukan dalam Tugas Akhir ini. Alur pemodelan merupakan rincian penjelasan berdasarkan blok diagram yang sudah dibuat sebelumnya. Alur ini terdiri dari:

1. Start, proses pemodelan dimulai.
2. Proses melakukan Studi Literatur, melakukan pengumpulan dan mempelajari literatur terkait MVVM, template method dan DI, dan *ck metrics* sebagai indikator *reusability*.
3. Proses melakukan Elisitasi , proses elisitasi yang digunakan berupa observasi buku panduan tugas akhir dan website tugas akhir sebagai sarana pengamatan mandiri.
4. Proses pemodelan artefak, penyusunan kebutuhan sistem dan membuat use case untuk diimplementasikan pada aplikasi S1 Informatika
5. Menganalisis ketepatan pemilihan *design pattern* terhadap *code smell* .Hal ini dilakukan agar *code smell* tidak terbawa pada aplikasi S1 Informatika
6. Daur Ulang base application.Melakukan modifikasi berdasarkan model yang telah
7. dibuat dan fragment code reuse yang ditetapkan
8. Testing dilakukan melalui dua proses ck metrics. Pada tahap ini mengukur rata-rata skor ck metrics pada aplikasi setelah dan sebelum modifikasi
9. Evaluasi hasil
10. Membuat kesimpulan
11. End. Proses pemodelan kegiatan selesai.



Gambar 5. Alur Pemodelan Kegiatan

4. Evaluasi

Requirement Statement

Require Statement merupakan pernyataan yang menginformasikan bagaimana keadaan sistem saat ini pada Aplikasi sidang tugas akhir S1 yang diperoleh dengan melakukan kegiatan elisitasi berupa observasi buku panduan sidang tugas akhir S1 Informatika. *Requirement Statement* terdiri dari dua bagian, yaitu *Functional Requirement* dan *Non-Functional Requirement*.

Functional Statement

Pada Functional Requirement memberikan informasi mengenai fungsi yang dilakukan oleh sistem dengan menyertakan deskripsi dari kegunaan fungsi pada sistem, Adapun masing-masing penjelasannya dapat dilihat dari Tabel 3.

Tabel 3. Functional Requirement

ID	Pengguna	Requirement Statement	Use Case Related	Priority
FR-01	Mahasiswa	Mahasiswa melihat status SK TA	UC01	Must
UC02		Mahasiswa dapat mengajukan form SK TA	UC01	Must

FR-03		Mahasiswa dapat melakukan bimbingan	UC03	Must
FR-04		Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur reguler	UC04,UC06	Must
FR-05		Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur reguler	UC04	Must
FR-06		Mahasiswa dapat melihat jadwal atau prediksi jadwal sidang terjadwal berdasarkan waktu dikeluarkannya SK TA	UC01	Must
FR-07		Mahasiswa dapat mengumpulkan lembar revisi	UC09	Must
FR-08	Dosen	Dosen memvalidasi Bimbingan	UC03	Must
FR-09		Dosen dapat memvalidasi form pendaftaran sidang reguler	UC04	Must
FR-10		Dosen pembimbing dan penguji dapat memberikan nilai sidang	UC04	Must
FR-11		Sistem menyediakan kalkulator nilai	UC04	Optional
FR-12		Dosen dapat memvalidasi form SK	UC08	Must
FR-13	Prodi	Admin prodi dapat mengunggah form excel plotting pembimbing dan mahasiswa serta membuat SK TA	UC01	Must
FR-14		Admin prodi dapat melihat daftar mahasiswa dan dosen pembimbing	UC01	Must
FR-15		Admin prodi dapat mempublish excel terkait jadwal sidang	UC04	Must

FR-16		Admin prodi melihat daftar nilai dan berita acara sidang	UC04	Must
F17		Admin prodi dan LAK melihat daftar status revisi peserta sidang	UC04	Must
FR-18		Mengelola data mahasiswa	UC02	Must
FR-19		Mengelola data dosen	UC03	MUST
FR-20	LAK	Admin LAK dapat mengatur periode pendaftaran sidang	UC05	Must
FR-21		Admin LAK melihat daftar mahasiswa peserta sidang	UC04	Must

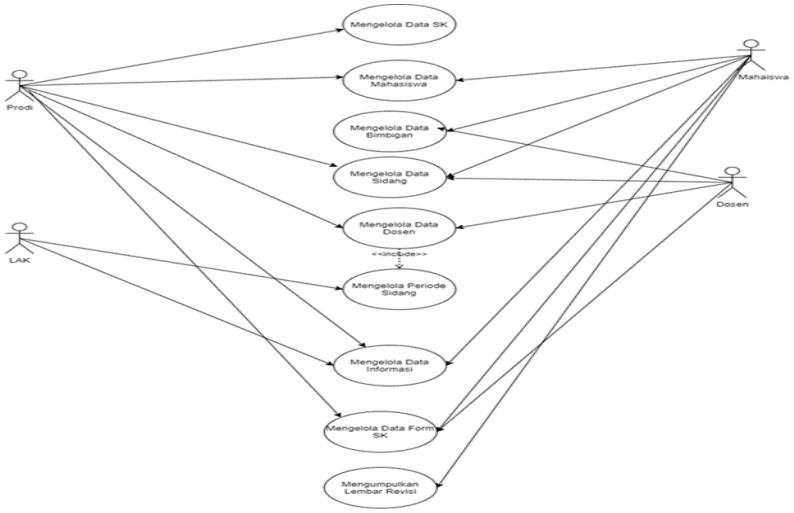
Non-Functional Statement

Tabel 4. Requirement Statement and NFR Types

ID	Requirement Statement and NFR Types
NF01	Pengguna dapat saling memvalidasi kegiatan yang berlangsung (Usability).
NF02	Pengguna harus dapat menggunakan sistem secara bersamaan (Reliability).
NF03	Sistem harus dapat diakses setiap saat oleh dosen, mahasiswa, dan prodi,LAK (Availability).

Pemodelan Sidang Akhir

Model UML digunakan untuk menggambarkan proses bisnis yang berlangsung pada proses sidang tugas akhir S1 Informatika . Use Case sendiri adalah gambaran fungsional dari sebuah sistem.Merujuk pada gambar 6 aktor yang terlibat adalah dosen ,mahasiswa , prodi dan lak



Gambar 6. Use Case

Identifikasi Base Application

Pada tahap ini dihasilkan dua buah hasil yaitu *fragment code reuse* dan hasil identifikasi kelas bermasalah dari alat bantu *code MR*.Tahap ini akan menjadi dasar dalam melakukan daur ulang atau modifikasi kode **.**

Fragment Code Reuse

Merujuk pada tabel 5 terdapat daftar fitur yang kita pilih untuk digunakan kembali . Dasar pemilihan ini dilakukan dengan dasar kemiripan dengan kebutuhan sistem sehingga dapat digunakan kembali.

Tabel 5. Tabel Code Reuse

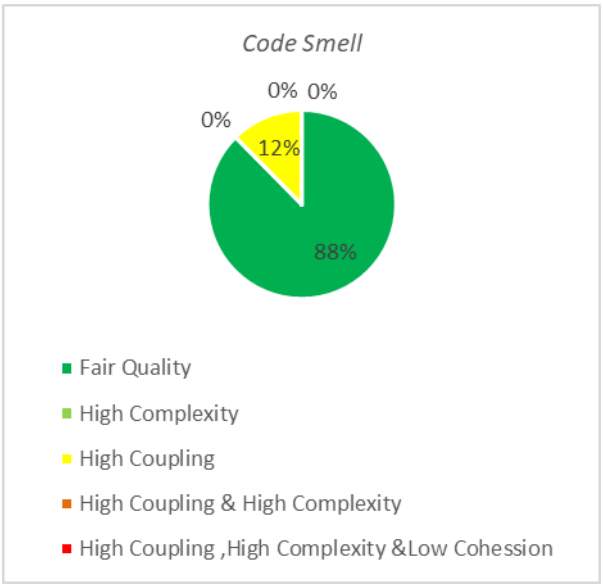
Pengguna	**Fitur**	**Deskripsi**	**Status**
Koordinator proyek akhir	Menambahkan informasi	fungsi ini digunakan untuk menambahkan informasi yang terkait dengan proses pelaksanaan proyek akhir.	Digunakan Kembali
	Menambahkan dosen	Fungsi ini digunakan untuk menambahkan dosen tetap D3 Rekayasa Perangkat Lunak Aplikasi.	Digunakan kembali
	Menambahkan mahasiswa	Fungsi ini digunakan untuk menambahkan mahasiswa D3 Rekayasa Perangkat Lunak Aplikasi yang melaksanakan proyek akhir.	Digunakan kembali
	Menambahkan kategori judul	Fungsi ini digunakan untuk menambahkan kategori judul yang nantinya kategori ini akan digunakan untuk menambahkan judul oleh dosen.	Tidak Digunakan
	Menambahkan kategori monitoring dan evaluasi	Fungsi ini digunakan untuk menambahkan kategori monitoring dan evaluasi yang akan digunakan untuk menambahkan monitoring dan evaluasi.	Tidak Digunakan
	Menambahkan judul dosen	Fungsi ini digunakan untuk menambahkan judul untuk dosen bila dosen yang bersangkutan ingin dibuatkan judul oleh koordinator Proyek Akhir.	Tidak Digunakan
	Pemetaan monitoring dan evaluasi	Fungsi ini digunakan untuk memetakkan	Tidak Digunakan

		kelompok proyek akhir dengan dosen reviewer untuk melaksanakan monitoring dan evaluasi.	
Dosen	Menambahkan informasi	fungsi ini digunakan untuk menambahkan informasi yang terkait dengan proses pelaksanaan proyek akhir	Digunakan kembali
	Menambahkan judul	fungsi ini digunakan untuk menambahkan judul untuk digunakan oleh mahasiswa sebagai judul proyek akhir.	Tidak Digunakan
	Menyetujui dan menolak judul dosen	Fungsi ini digunakan untuk menerima ataupun menolak judul yang ajukan oleh kelompok berdasarkan judul yang diajukan dosen.	Tidak Digunakan
	Menyetujui dan menolak judul mandiri	Fungsi ini digunakan untuk menerima ataupun menolak judul yang ajukan oleh kelompok berdasarkan judul yang diajukan secara mandiri.	Tidak Digunakan
	Menyetujui bimbingan	Fungsi ini digunakan dosen untuk menyetujui dengan hasil input bimbingan yang dilakukan oleh mahasiswa.	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
	Menambahkan nilai sidang	Fungsi ini digunakan dosen untuk menambahkan nilai dan review sidang untuk masing-masing mahasiswa.	Tidak Digunakan
Mahasiswa	Mengajukan judul dosen	Fungsi ini digunakan oleh mahasiswa untuk memilih judul dari dosen.	Tidak Digunakan
	Mengajukan judul mandiri	Fungsi ini digunakan oleh mahasiswa untuk	Tidak Digunakan

		mengajukan judul secara mandiri kepada dosen yang ingin dijadikan pembimbing.	
	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
General	Ubah profil	Fungsi ini digunakan oleh pengguna untuk mengubah profil	Digunakan kembali

Code Smell

Pada tahap ini saya akan mendeteksi apakah base application yang telah kita pilih memiliki code smell atau tidak. Untuk melakukan hal tersebut kita akan menggunakan alat bantu pengukuran yang bernama *CodeMR* . *Tools* ini menghasilkan beberapa object-oriented metric, salah satunya *CK Metrics* yang terdiri dari CBO, DIT, WMC, NOC, RFC, LCOM. Serta alat ini mendeteksi code smell berdasarkan 4 atribut kualitas perangkat lunak yaitu *coupling*,*lack of cohesion*, *complexity*, dan *size*.Seperti yang ditunjukan pada gambar 4.4.1 terdapat 12% komponen pada aplikasi *base application* yang memiliki masalah pada tingkat coupling untuk kelas apa saja yang dapat dilihat lebih detail pada lampiran 1



Gambar 7. Code Smell Base Application

Analisa Ketepatan *Design Pattern*

Setelah menetapkan *fragment code* yang akan di daur ulang, tahapan selanjutnya adalah melakukan validasi terkait penerapan *design pattern* yang digunakan. Peneliti dalam menentukan implementasi *design pattern* mengacu pada Buku Design pattern: *Elements of Reusable Object-Oriented Software* [5], terdapat enam cara dalam memilih *design pattern*, yaitu:

1. Pertimbangkan bagaimana design pattern memecahkan masalah design.
2. Analisis pada bagian Intent setiap design pattern.
3. Pelajari bagaimana pola saling berhubungan.
4. Pola studi dengan tujuan yang sama.

5. Periksa penyebab *redesign.*
6. Pertimbangkan apa yang harus menjadi variabel dalam desain .

Berdasarkan paparan tersebut peneliti memilih pendekatan nomor 5 dengan melakukan identifikasi *code smell* untuk menentukan penyebab *redesign*.Berdasarkan penjelasan pada buku tersebut terdapat delapan alasan mengapa harus melakukan *redesign* yaitu :

1. Membuat objek dengan menentukan kelas secara eksplisit.
2. Dependensi terhadap operasi tertentu.
3. Dependensi terhadap platform perangkat keras dan perangkat lunak.
4. Dependensi terhadap representasi sebuah objek atau implementasi.
5. Dependensi terhadap algoritma.
6. Tight Coupling (High Coupling).
7. Memperluas fungsionalitas dengan membuat subclass.
8. Kesulitan untuk mengubah sebuah kelas dengan mudah.

Dari delapan alasan untuk melakukan redesign sebuah sistem, terdapat dua faktor penyebab yang ditemukan pada studi kasus ini, yaitu alasan nomor enam dan tujuh. Pada penelitian ini sebelumnya telah dipaparkan 3 metode untuk meningkatkan *reusability* .Pada tahapan ini akan menjelaskan bagaimana 3 metode tersebut mengatasi masalah pada point nomor 6 dan 7.

Merujuk pada tabel dibawah ini akan menjelaskan alasan untuk mengatasi masalah point nomor 6 dan 7 dan untuk meningkatan kualitas atribut Ck metrics pada atribut ck metrics

Tabel 6. Analisis *Design Pattern*

Design Problem/Peningkatan atribut Ck Metrics	Bagian Kode	Design Pattern	Alasan Pemilihan
1. Tight Coupling (High Coupling) 2. Menurunkan Nilai CBO.	Pada kelas activity fragment ,viewmodel, repo	DI & MVVM	1. DI juga dapat menurunkan CBO sebesar 10%[16]. 2. Metode MVVM memiliki atribut data binding untuk mengurai coupling dan komponen pada MVVM memiliki tingkat coupling yang rendah[11][12][13]
Menurunkan atribut LCOM	Pada kelas activity fragment ,viewmodel, repo	DI & MVVM	1. MVVM membagi fokus layer aplikasi lebih jelas 2. Pada MVVM kelas view tidak perlu mengimplementasikan antarmuka metode di view yang memerlukan parameter atau nilai Kembali [11][12][13] 3. DI juga
Menurunkan atributWMC	Pada kelas activity fragment ,viewmodel , repo	DI ,MVVM & Template Method	1. DI Tidak perlu melakukan inisialisasi objek sendiri dapat mengurangi ketergantungan 2. Berkat adanya template method tidak perlu membuat fungsi umum sendiri[5] 3. MVVM membagi fokus masalah aplikasi lebih jelas sehingga kompleksitas lebih rendah

1. Memperluas fungsionalitas dengan membuat subclass 2. Meningkatkan atribut DIT dan NOC	Pada kelas serupa dalam kasus ini adapter,viewholder,	Template Method	1. Setiap satu kelas activity , fragment,view holder ,adapter akan menambahkan skor DIT[5] 2. Template method yang membuka ruang untuk mendefinisikan ulang metode dari kelas abstrak hal ini akan dapat meningkatkan nilai NOC[5] 3. Kelas dengan template method dapat digunakan berbagai tempat[5]

Modifikasi Base Application

Setelah menentukan bagaimana metode yang dipilih untuk diterapkan, selanjutnya dilakukan modikasi *base application* dengan mengubah beberapa kode agar sesuai dengan kebutuhan sistem yang baru, serta menghapus baris kode yang sekiranya tidak dibutuhkan dalam sistem. Tahapan daur ulang yang akan dilalui pada penelitian kali ini adalah sebagai berikut :

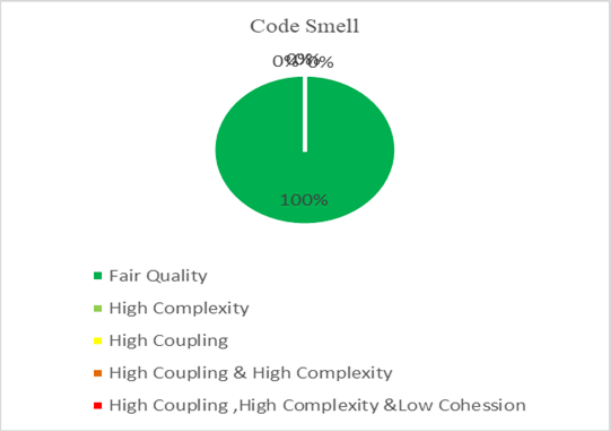
1. Mengambil bagian kode yang diperlukan berdasarkan fragment code reuse yang telah ditentukan
2. Melakukan penerapan 3 metode yang telah dipilih
3. Melakukan modifikasi agar sesuai dengan kebutuhan sistem yang telah ditentukan
4. Melakukan pembaharuan kepada versi *library* yang digunakan jika ada versi terbaru
5. Selesai

Kendala dan kemudahan yang peneliti dapat selama melakukan proses diatas dapat dilihat pada lampiran 9.Setelah mengimplementasikan 3 metode yang ditentukan dan juga melakukan pembaruan terhadap versi web service yang semula menggunakan Laravel versi 5.6 ke Laravel versi 8.Diharapkan aplikasi hasil daur ulang memiliki tingkat *reusability* yang lebih baik dibandingkan dengan kondisi sebelumnya. Selain itu juga proses ini bertujuan untuk mengatasi *code smell* yang ada.

Testing

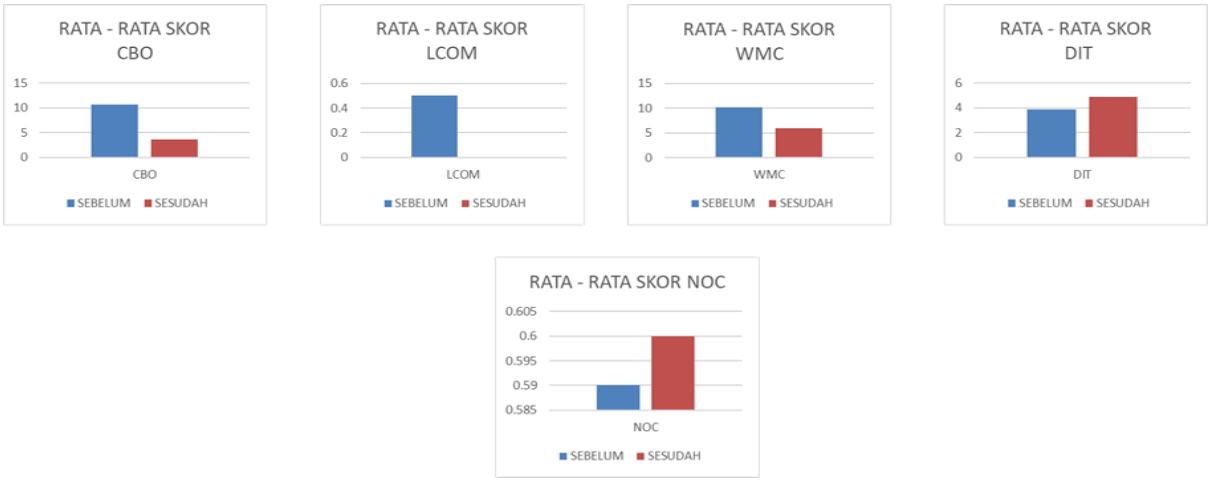
Testing Reusability

Sebelum melakukan pengukuran reusability dengan *ck metrics* akan dilakukan identifikasi *code smell* pada aplikasi daur ulang.Hal ini dilakukan agar masalah yang sama tidak terulang lagi. Merujuk pada gambar 4.6.1 aplikasi daur ulang tidak memiliki kelas bermasalah sama sekali.Selain itu juga telah dilakukan proses validasi tools seperti lampiran 2



Gambar 8. Code Smell Base Application

Pengukuran reusability dengan ck metrics dengan alat bantu code mr. Pada tahap ini akan mengukur ck metrics terkait reusability. Merujuk pada gambar 4,6,2 menjelaskan perbandingan rata-rata skor ck metrics sebelum dan sesudah refactoring. CBO , LCOM dan mengalami penurunan sesuai diharapkan. DIT dan NOC mengalami peningkatan . Penurunan CBO ,LCOM dan WMC menjadi indikasi reusability yang lebih baik. Peningkatan NOC
dan DIT juga menjadi indikasi reusability yang lebih baik.



Gambar 9. CK Metrics Testing

Testing Skenario

Pada tahapan ini mengkonfirmasi skenario yang akan dilalui pengguna . Tahapan ini dilakukan dengan melakukan video conference dengan perwakilan 3 skenario yang akan dilaksana dapat dilihat pada tabel dibawah ini

Tabel 6. Task Testing Plot SK

Plot Sk	
NO	TAHAPAN
1	Prodi mengklik menu sk
2	Prodi melihat daftar sk aktif dan nonaktif
3	Prodi mengklik menu tambah
4	Prodi upload file excel sk

Tabel 7. Task Testing Plot SK

Perpanjang Sk	
NO	TAHAPAN
1	Mahasiswa Melakukan pengajuan perpanjang SK
2	Dosen Melakukan persetujuan
3	Prodi Melakukan persetujuan
4	Mahasiswa dapat melihat sk sudah diperpanjang

Tabel 8. Skenario Sidang

Sidang	
NO	TAHAPAN
1	Admin Lak menambahkan Periode pendaftaran Sidang
2	Mahasiswa Daftar sidang
3	Prodi Melihat daftar sidang
4	Prodi Memplotting sidang
5	Mahasiswa Dapat Melihat jadwal Sidang nya
6	Dosen memberikan Nilai Sidang
7	Semua Pihak yang terlibat Melihat Berita Acara

Pada tahapan ini juga dihasilkan beberapa saran fitur tambahan sebagai berikut :

1. Download excel berita acara sidang bagi LAK
2. Format Plotting SK disesuaikan dengan prodi
3. Dosen pembimbing 1 dapat memberi catatan revisi

Dampak testing scenario yang dilakukan kepada user terhadap reusability dapat dilihat pada tabel dibawah ini :

Tabel 9. Tabel Analisis Daur Ulang Setelah Mendapatkan Input Dari User

NO	Indikator	Nilai
1	Kelas Bermasalah	0
2	CK Metrics	CK Metrics Daur Ulang Pertama CBO = 3,60 LCOM = 0 DIT = 4,87 NOC=0,60 WMC = 6,01 CK Metrics Daur Ulang Kedua CBO = 3,60 LCOM = 0 DIT = 4,67 NOC=0,60 WMC = 6,19

Merujuk tabel 9 nilai metrics yang dintunjukan tidak terlalu mengalami perubahan yang berarti nilai hampir sama.Jumlah kelas bermasalah juga tetap masih belum ada pada saat daur ulang kedua. Maka oleh dari itu dapat.Berdasarkan hal ini dapat disimpulkan bahwa metode yang telah digunakan dapat mempertahankan nilai kualitasnya walaupun ada perubahan

5. Kesimpulan

Berdasarkan hasil dan pembahasan pada penelitian ini, terdapat kesimpulan mengenai Implementasi Pola Arsitektur Model-view-Viewmodel Untuk Reusability Perangkat Lunak, yaitu:

1. Terdapat kegiatan elisitasi kebutuhan yang telah menghasilkan FR dan NON FR. Untuk FR adalah sebagai berikut :
    1. Mahasiswa melihat status SK TA
    2. Mahasiswa dapat mengajukan form SK TA
    3. Mahasiswa dapat melakukan bimbingan
    4. Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur reguler
    5. Mahasiswa mendaftarkan diri ke sidang tugas akhir jalur terjadwal
    6. Mahasiswa dapat melihat jadwal atau prediksi jadwal siding terjadwal berdasarkan waktu dikeluarkannya SK TA
    7. Mahasiswa dapat mengumpulkan lembar revisi

8. Dosen memvalidasi Bimbingan
9. Dosen dapat memvalidasi form pendaftaran sidang reguler
10. Dosen pembimbing dan penguji dapat memberikan nilai sidang
11. Sistem menyediakan kalkulator nilai
12. Dosen dapat memvalidasi form SK
13. Admin prodi dapat mengunggah form excel plotting pembimbing dan mahasiswa serta membuat SK TA
14. Admin prodi dapat melihat daftar mahasiswa dan dosen pembimbing
15. Admin prodi dapat mempublish excel terkait jadwal sidang
16. Admin prodi melihat daftar nilai dan berita acara sidang
17. Admin prodi dan LAK melihat daftar status revisi peserta sidang
18. Mengelola data mahasiswa
19. Mengelola data dosen
20. Admin LAK dapat mengatur periode pendaftaran sidang
21. Admin LAK melihat daftar mahasiswa peserta sidang.

Untuk NFR adalah sebagai berikut :

1. Pengguna dapat saling memvalidasi kegiatan yang berlangsung (Usability).
2. Pengguna harus dapat menggunakan sistem secara bersamaan (Reliability).
3. Sistem harus dapat diakses setiap saat oleh dosen, mahasiswa, dan prodi,LAK (Availability).

2. Berdasarkan FR dan NFR, selanjutnya pada penelitian ini telah berhasil dirancang pemodelan artefak UML menggunakan Use Case Diagram. Pada artefak ini terdiri dari aktor dan use case. Untuk aktornya terdiri dari prodi lak,dosen dan mahasiswa. Untuk Use Case terdiri dari mengelola data sk, mengelola data mahasiswa, mengelola data bimbingan, mengelola data sidang, mengelola data dosen, mengelola data informasi dan mengelola data form sk
3. Selanjutnya dilakukan identifikasi untuk aplikasi D3 berdasarkan requirement statement dan artefak yg telah dirancang. Hal ini harus dilakukan untuk menentukan fragment code reuse berdasarkan kemiripan pada model yang telah dibuat dan mendeteksi code smell. Pada fragment code reuse terdapat daftar fitur yang akan didaur ulang Selain itu, telah menghasilkan diagram kelas dan tabel kelas bermasalah(*code smell*)
4. Merujuk hasil perancangan artefak UML, fragment code, code smell. Telah dihasilkan kegiatan proses daur ulang kode implementasi MVVM, template method, dan dependency injection berupa aplikasi baru dan hasil identifikasi code smell
5. Telah menghasilkan atribut ck metrics terkait reusability, yaitu hasil pengukuran untuk skor CBO,LCOM,WMC DIT dan NOC . Terdapat dampak implementasi dari metode MVVM, *template method*, dan *dependency injection* terhadap *reusability*, berupa/yaitu penurunan skor pada atribut CBO, LCOM serta WMC dan peningkatan skor pada atribut DIT serta NOC
6. Untuk penelitian selanjutnya peneliti dapat menggunakan pemrograman java karena lebih menerapkan konsep pemrograman berbasis objek dibanding bahasa kotlin atau dart

Daftar Pustaka

[1] "Why Code Reuse Matters." https://www.apress.com/de/blog/all-blog-posts/why-code-reuse-matters/15477476 (accessed Jun. 20, 2021).

[2] J. L. Leal, J. P. Rodríguez, and O. A. Gallardo, "Project time: Time management method for software development projects-analytical summary," *J. Phys. Conf. Ser.*, vol. 1126, no. 1, 2018, doi: 10.1088/1742-6596/1126/1/012030.

[3] S. Younoussi and O. Roudies, "All about software reusability: A sistematic literature review," *J. Theor. Appl. Inf. Technol.*, vol. 76, no. 1, pp. 64–75, 2015.

[4] A. Mateen, S. Kausar, and A. R. Sattar, "A Software Reuse Approach and Its Effect On Software Quality, An Empirical Study for The Software Industry," vol. 7, no. 2, pp. 266–279, 2017, [Online]. Available: http://arxiv.org/abs/1702.00125.

[5] M. Hitz, "Oois' 95," *Oois' 95*, no. February, 1996, doi: 10.1007/978-1-4471-1009-5.

[6] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, 1994, doi: 10.1109/32.295895.
[7] S. Bose, "a Comparative Study: Java Vs Kotlin Programming in Android Application Development," *Int. J. Adv. Res. Comput. Sci.*, vol. 9, no. 3, pp. 41–45, 2018, doi: 10.26483/ijarcs.v9i3.5978.
[8] M. H. M. and N. A. N., "Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design," *APTIKOM J. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 63–76, 2016, doi: 10.34306/csit.v1i2.49.
[9] P. Mago, J. & Kaur, "Analysis of quality of the design of the object oriented software using fuzzy logic," *Int. Conf. Recent Adv. Futur. Trends Inf. Technol. Proc. Publ. Int. J. Comput. Appl.*, pp. 21–25, 2012, [Online]. Available: https://pdfs.semanticscholar.org/043f/ddd25c3af905a24d762ee832221170f4bcb7.pdf.
[10] X. Li, D. Chang, H. Pen, X. Zhang, Y. Liu, and Y. Yao, "Application of MVVM design pattern in MES," *2015 IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. IEEE-CYBER 2015*, no. 2012, pp. 1374–1378, 2015, doi: 10.1109/CYBER.2015.7288144.
[11] "A comparison of Android Native App Architecture Master ' s Programme in ICT Innovation A Comparison of Android Native App Architecture – MVC , MVP and MVVM," 2016.
[12] F. Sholichin, M. A. Bin Isa, S. A. Halim, and M. F. Bin Harun, "Review of ios architectural pattern for testability, modifiability, and performance quality," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 15, pp. 4021–4035, 2019.
[13] A. Syromiatnikov and D. Weyns, "A journey through the land of model-view-design patterns," *Proc. - Work. IEEE/IFIP Conf. Softw. Archit. 2014, WICSA 2014*, pp. 21–30, 2014, doi: 10.1109/WICSA.2014.13.
[14] "Panduan arsitektur aplikasi | Developer Android | Android Developers." https://developer.android.com/jetpack/guide (accessed Jun. 20, 2021).
[15] M. T. H. E. State and L. S. Technology, "Usage of Dependency Injection within different frameworks," no. March, 2020.
[16] E. Razina and D. Janzen, "Effects of dependency injection on maintainability," *Proc. 11th IASTED Int. Conf. Softw. Eng. Appl. SEA 2007*, pp. 7–12, 2007.
[17] E. Gamma;Richard Helm;Ralph E. Johnson; J. Vlissides, *Design patterns: elements of reuseable object-oriented software*. 1994.
[18] N. Qamar and A. A. Malik, "Impact of Design Patterns on Software Complexity and Size," *Mehran Univ. Res. J. Eng. Technol.*, vol. 39, no. 2, pp. 342–352, 2020, doi: 10.22581/muet1982.2002.10.
[19] S. Hussain, J. Keung, and A. A. Khan, "The effect of gang-of-four design patterns usage on design quality attributes," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, no. July, pp. 263–273, 2017, doi: 10.1109/QRS.2017.37.
[20] J. Kouraklis, "MVVM in delphi: Architecting and building model view viewmodel applications," *MVVM Delphi Archit. Build. Model View ViewModel Appl.*, no. October 2016, pp. 1–143, 2016, doi: 10.1007/978-1-4842-2214-0.
[21] S. Rochimah, P. G. Nuswantara, and R. J. Akbar, "Analyzing the Effect of Design Patterns on Software Maintainability: A Case Study," *2018 Electr. Power, Electron. Commun. Control. Informatics Semin. EECCIS 2018*, pp. 326–331, 2018, doi: 10.1109/EECCIS.2018.8692876.
[22] I. Binanto, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman, and B. Soewito, "Measuring the quality of various version an object-oriented software utilizing CK metrics," *2018 Int. Conf. Inf. Commun. Technol. ICOIACT 2018*, vol. 2018-Janua, pp. 41–44, 2018, doi: 10.1109/ICOIACT.2018.8350760.
[23] N. Cherdsakulwong and T. Suwannasart, "Impact Analysis of Test Cases for Changing Inputs or Outputs of Functional Requirements," *Proc. - 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2019*, pp. 179–183, 2019, doi: 10.1109/SNPD.2019.8935754.
[24] S. Hussain, J. Keung, and A. A. Khan, "The effect of gang-of-four design patterns usage on design quality attributes," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 263–273, 2017, doi: 10.1109/QRS.2017.37.
[25] F. Khomh and Y. G. Guéhéneuc, "Do design patterns impact software quality positively?," *Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR*, pp. 274–278, 2008, doi: 10.1109/CSMR.2008.4493325.
[26] S. Roubtsov, A. Serebrenik, and M. Van Den Brand, "Detecting modularity 'smells' in dependencies injected with Java annotations," *Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR*, pp. 244–247, 2010, doi: 10.1109/CSMR.2010.45.
[27] A. Ampatzoglou, A. Chatzigeorgiou, S. Charalampidou, and P. Avgeriou, "The effect of GoF design patterns on stability: A case study," *IEEE Trans. Softw. Eng.*, vol. 41, no. 8, pp. 781–802, 2015, doi: 10.1109/TSE.2015.2414917.

[28] Y. Lim, M. Kim, S. Jeong, and A. Jeong, "A reuse-based software development method," *Proc. - 2008 Int. Conf. Converg. Hybrid Inf. Technol. ICHIT 2008*, pp. 102–109, 2008, doi: 10.1109/ICHIT.2008.190.
[29] S. Jiang and H. Mu, "Design patterns in object oriented analysis and design," *ICSESS 2011 - Proc. 2011 IEEE 2nd Int. Conf. Softw. Eng. Serv. Sci.*, pp. 326–329, 2011, doi: 10.1109/ICSESS.2011.5982229.
[30] P. Pradhan, A. K. Dwivedi, and S. K. Rath, "Impact of Design Patterns on Quantitative Assessment of Quality Parameters," *Proc. - 2015 2nd IEEE Int. Conf. Adv. Comput. Commun. Eng. ICACCE 2015*, pp. 577–582, 2015, doi: 10.1109/ICACCE.2015.102.
[31] T. Mikkonen and A. Taivalsaari, "Software reuse in the era of opportunistic design," *IEEE Softw.*, vol. 36, no. 3, pp. 105–111, 2019, doi: 10.1109/MS.2018.2884883.

Lampiran

Lampiran 1 Tabel *Code Smell* dan kelas bermasalah

No	Nama Kelas	Coupling	CBO
1	KoorMainActivity	*Very-High*	38
2	DosenSidangActivity	*High*	27
3	MahasiswaJudulPaMandiriPengajuanActivity	*High*	28
4	MahasiswaPaFragment	*High*	23
5	DosenSidangUbahActivity	*High*	23
6	KoorPemetaanMonevDetailActivity	*High*	24
7	DosenSidangTambahActivity	*High*	22
8	DosenProyekAkhirActivity	*High*	26
9	MahasiswaJudulPaDosenPengajuanActivity	*High*	23
10	KoorProyekAkhirDetailActivity	*High*	27
11	DosenJudulPaSubdosenAccActivity	*High*	22
12	DosenJudulPaSubmahasiswaAccActivity	*High*	23
13	DosenMonevTambahActivity	*High*	24
14	DosenJudulPaSubdoDetailActivity	*High*	27
15	DosenProyekAkhirBimbinganActivity	*High*	22
16	DosenJudulPaSubdosenUbahActivity	*High*	21
17	MahasiswaMainActivity	*High*	23
18	DosenMainActivity	*High*	24
19	KoorJudulPaSubdosenUbahActivity	*High*	21
20	MahasiswaJudulPaArsipActivity	*High*	22
21	DosenJudulPaArsipActivity	*High*	22
22	DosenMonevMahasiswwaActivity	*High*	23
23	MahasiswaPaBimbinganActivity	*High*	23
24	MahasiswaProfilUbahActivity	*High*	23
25	KoorProfilUbahActivity	*High*	22
26	MahasiswaPaBimbinganTambahanActivity	*High*	22
27	DosenProfilUbahActivity	*High*	23
28	MahasiswaPaSidangActivity	*High*	21
29	KoorDosenDetailActivity	*High*	24
30	KoorMahasiswaDetailActivity	*High*	24
31	MahasiswaPaMonevDetailActivity	*High*	22
32	DosenMonevDetailActivity	*High*	22
33	DosenInformasiDetailActivity	*High*	25

Lampiran 2 Tabel Pengkategorian

NO.	Aspek	Pengkategorian
1	CBO	$0 \leq CBO < 5 = LOW$ $5 \leq CBO < 10 = LOW - MEDIUM$ $10 \leq CBO < 20 = MEDIUM - HIGH$ $20 \leq CBO < 30 = HIGH$ $CBO \geq 30 = VERY - HIGH$
2	LCOM	$LCOM < 0.5 = LOW$ $0.5 \leq LCOM < 0.7 = LOW - MEDIUM$ $0.7 \leq LCOM < 0.9 = MEDIUM - HIGH$ $0.9 \leq LCOM < 1 = HIGH$ $LCOM \geq 1 = VERY - HIGH$
3	WMC	$0 \leq WMC < 20 = LOW$ $20 \leq WMC < 50 = LOW - MEDIUM$ $50 \leq CBO < 101 = MEDIUM - HIGH$ $101 \leq WMC < 120 = HIGH$ $WMC \geq 12O = VERY - HIGH$
4	DIT	$0 \leq DIT < 1 = LOW$ $1 \leq DIT < 3 = LOW - MEDIUM$ $3 \leq DIT < 10 = MEDIUM - HIGH$ $10 \leq DIT < 20 = HIGH$ $DIT \geq 2O = VERY - HIGH$
5	NOC	$0 \leq NOC < 1 = LOW$ $1 \leq NOC < 5 = LOW - MEDIUM$ $5 \leq NOC < 10 = MEDIUM - HIGH$ $10 \leq NOC < 15 = HIGH$ $NOC \geq 15 = VERY - HIGH$

Lampiran 3 Tabel Validasi Tools

NO	Nama Kelas	CBO		LCOM	
		Manual	CodeMr	Manual	CodeMr
1	Signin Activity	Coupling dalam kelas: 1. Bundle 2. Data bindingUtil 3. authViewModel 4. Signin Total = 4	4	LCOM =0 karena hanya ada satu variabel dalam sebuah kelas	0

2	Mahasiswa Bimbingan Activity	Coupling dalam kelas: 1. Bundle 2. Data bindingUtil 3. authViewModel 4. Signin 5. Menu Item Total = 5	5	m=5 a=14 m[dsnNip]=3 m[mhsNim]=3 m[dsnName]=2 m[adapter]=2 m[bimbinganViewModel]=2 m[USER_SP]=1 m[DOSEN_SP]= 0 m[PRODI_SP]=0 m[LAK_SP]=0 m[MAHASISWA_SP]=0 m[user]=2 m[progressDialog]=2 m[binding]=1 m[this]=3 sum(mA)=35 LCOM = (m - sum(mA)/a) / (m-1) = (5-(35/5))/(5-1)=-0.5 Batas LCOM $0 \leq$ LCOM ≤ 2 Hasil akhir = 0	0
3	Dosen Informasi DetailActivity	Coupling dalam kelas: 1. Bundle 2. Data bindingUtil 3. InformasiViewModel 4. Informasi 5. Menu 6. Menu Item Total = 6	6	m=5 a= 11 m[informasi]=3 m[informasiViewModel]=2 m[USER_SP]=1 m[DOSEN_SP]= 0 m[PRODI_SP]=0 m[LAK_SP]=0 m[MAHASISWA_SP]=0 m[user]=1 m[progressDialog]=2 m[binding]=1 m[this]=3 sum(mA)=24 LCOM = (m - sum(mA)/a) / (m-1) = (5-(24/5))/(5-1)=0.05	0

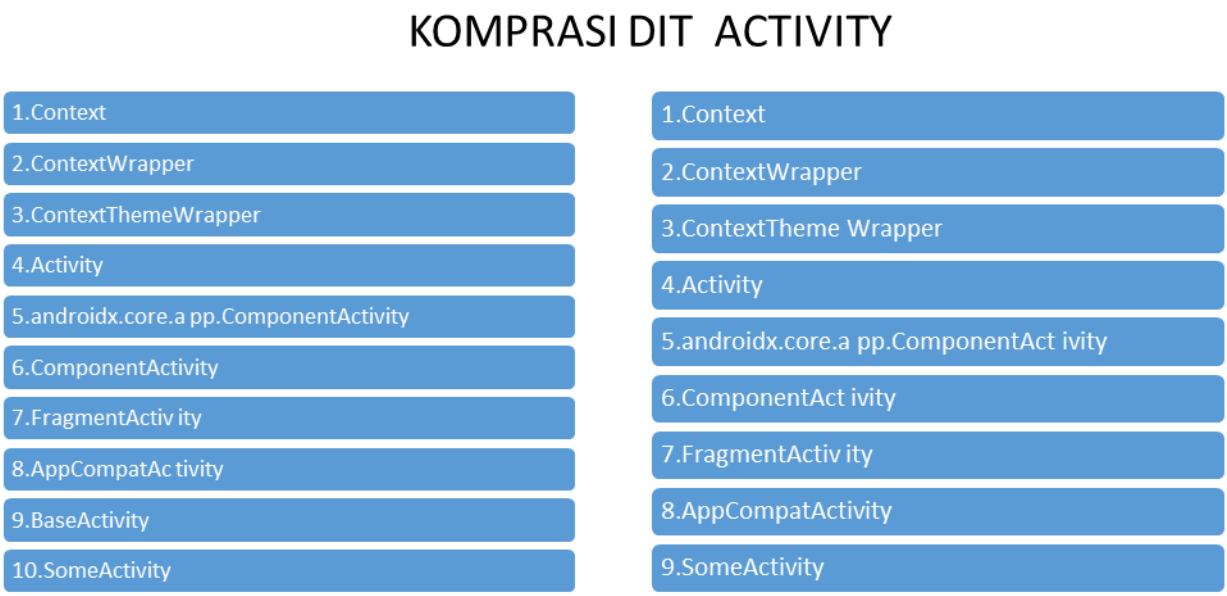
NO	Nama Kelas	WMC		DIT	
		Manual	CodeMr	Manual	CodeMr
1	Signin Activity	1. onCreate Edge=10 Node=10 C1 = Edge-Node+2=10-10+2=2	10	Kedalaman kelas dari root: 1. Context 2. ContextWrapper 3. ContextTheme	10

		2. Login Edge=8 Node=9 C3 = Edge-Node+2=8-9+2=1 3. getAuthViewModel Edge=0 Node=1 C4= Edge-Node+2=0-1+2=1 4. goToMainPage Edge=16 Node=17 C5= Edge-Node+2=16-17+2=1 5. succesSignin Edge=13 Node=10 C5= Edge-Node+2=13-10+2=5 WMC = C1+ C2 + C3 +C4+ C5= 10		Wrapper 4. Activity 5. androidx.core.app. ComponentActivity 6. ComponentActivity 7. FragmentActivity 8. AppCompatActivity 9. BaseActivity 10. SigninActivity Total = 10	
2	Mahasiswa Bimbingan Activity	1. onCreate Edge=13 Node=14 C1= Edge-Node+2=13-14+2=1 2. onOptionsItem Selected Edge=7 Node=8 C3 = Edge-Node+2=1 3. observe Edge=8 Node=9 C4= Edge-Node+2=8-9+2=1	5	Kedalaman kelas dari root: 1. Context 2. ContextWrapper 3. ContextThemeWrapper 4. Activity 5. androidx.core.app. ComponentActivity 6. ComponentActivity 7. FragmentActivity 8. AppCompatActivity 9. BaseActivity	10

		4. observe Edge=8 Node=9 C4= Edge-Node+2=8-9+2=1 5. startBimbingan Tambah Activity Edge=4 Node=5 C4= Edge-Node+2=5-4+2=1 WMC = C1+ C2 + C3 +C4+ C5 =5		10. MahasiswaBimbingan Activity Total = 10	
3	DosenInformasiDetailActivity	1. onCreate Edge=13 Node=14 C1 = Edge-Node+2=13-14+2=1 2. onCreateOptionsMenu Edge=0 Node=1 C2 = Edge-Node+2=0-1+2=1 3. onOptionsItemSelected Edge=7 Node=8 C3 = Edge-Node+2=7-8+2=1 4. getInformasiViewModel Edge=0 Node=1 C4= Edge-Node+2=0-1+2=1 5. deleteInformasi Edge=9 Node=10 C5= Edge-Node+2=9-10+2=1 WMC = C1+ C2 + C3 +C4+ C5 =5	5	Kedalaman kelas dari root: 1. Context 2. ContextWrapper 3. ContextThemeWrapper 4. Activity 5. androidx.core.app.ComponentActivity 6. ComponentActivity 7. FragmentActivity 8. AppCompatActivity 9. BaseActivity 10. DosenInformasiDetailActivity Total = 10	10

NO	Nama Kelas	NOC	
		Manual	CodeMr
1	Signin Activity	Kelas tidak melakukan *override* dari *baseactivity*	0
2	MahasiswaBimbingan Activity	Kelas tidak melakukan *override* dari *baseactivity*	0
3	DosenInformasiDetailActivity	Kelas tidak melakukan *override* dari *baseactivity*	0

Lampiran 4 Tabel Komporasi DIT



Lampiran 5 Komprasi Kode Presenter Dan ViewModel

Kode Singin Pada ViewModel

```
fun signIn(username: String, password: String) = repo.signin(username, password).asLiveData()
fun logout(token: String) = repo.logout(token).asLiveData()
```

Kode Login Pada Presenter

```
public void getLogin(String username, String password){
    if (connectionHelper.isConnected(context)){
        view.showProgress();
        ApiService apiInterface = ApiClient.getApiClient().create(ApiService.class);
        Call<User> call = apiInterface.setLogin(username, password);
        call.enqueue(new Callback<User>() {
            @Override
            public void onResponse(Call<User> call, Response<User> response) {
                view.hideProgress();
                if (response.isSuccessful() && response.body() != null){
                    boolean success = response.body().getSuccess();
                    if (success) {
                        view.onRequestSuccess(response.body());
                    } else {
                        view.onFailed(response.body().getMessage());
                    }
                }
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {
                view.hideProgress();
                view.onFailed(t.getLocalizedMessage());
```

Lampiran 6 Komparasi Penggunaan DI

Proses Pembentukan Api Service Pada NON DI

```
ublic void getDosen(){

   if (connectionHelper.isConnected(context)){
       viewResult.showProgress();

       ApiService apiInterface =
piClient.getApiClient().create(ApiService.class);
       Call<List<Dosen>> call = apiInterface.getDosen();
       call.enqueue(new Callback<List<Dosen>>() {
           @Override
           public void onResponse(Call<List<Dosen>> call,
esponse<List<Dosen>> response) {
               viewResult.hideProgress();
               if (response.body() != null && response.isSuccessful()) {
                   viewResult.onGetListDosen(response.body());
               } else {
                   viewResult.isEmptyListDosen();
               }

           }

           @Override
           public void onFailure(Call<List<Dosen>> call, Throwable t) {
               viewResult.hideProgress();
               viewResult.onFailed(t.getMessage());
           }
       });
   } else {
       Toast.makeText(context,
ontext.getString(R.string.validate_no_connection),
oast.LENGTH_SHORT).show();
   }




ublic void createDosen(String nip, String nama, String kode){

   if (connectionHelper.isConnected(context)){
       viewEditor.showProgress();

       ApiService apiInterfaceDosen =
piClient.getApiClient().create(ApiService.class);
       Call<Dosen> call = apiInterfaceDosen.createDosen(nip,nama,kode);
       call.enqueue(new Callback<Dosen>() {
           @Override
           public void onResponse(Call<Dosen> call, Response<Dosen>
esponse) {
               viewEditor.hideProgress();
               viewEditor.onSucces();
           }

           @Override
           public void onFailure(Call<Dosen> call, Throwable t) {
               viewEditor.hideProgress();
```

```
                viewEditor.onFailed(t.getLocalizedMessage());
            }
        });
    } else {
        Toast.makeText(context,
context.getString(R.string.validate_no_connection),
Toast.LENGTH_SHORT).show();
    }


}
```

Proses Pembentukan Api Service Pada DI

```
class AuthRepo @Inject constructor(
    private val apiService: ApiService,
) {


    fun signin(username: String, password: String) = flow<Resource<Signin>> {
        emit(Resource.Loading())
        try {
            val signin = apiService.signin(username, password)
            emit(Resource.Success(signin.data))
        } catch (throwable: Throwable) {
            emit(Resource.Error(throwable = throwable))
        }
    }


    fun logout(token: String) = flow<Resource<User>> {
        emit(Resource.Loading())
        try {
            val logout = apiService.logout(token)
            emit(Resource.Success(logout.data))
        } catch (throwable: Throwable) {
            emit(Resource.Error(throwable = throwable))
        }
    }


}
```

Lampiran 6 Komparasi Penggunaan Data Binding

Tanpa Databinding

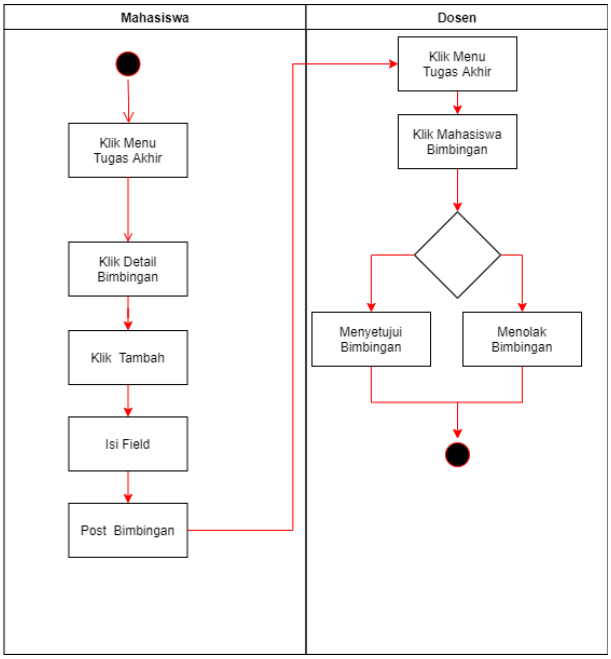
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    Button button_login = findViewById(R.id.act_main_button_login);
    editTextUsername = findViewById(R.id.act_main_edittext_username);
    editTextPassword = findViewById(R.id.act_main_edittext_password);

    sessionManager = new SessionManager(this);
    loginPresenter = new LoginPresenter(this);
    loginPresenter.initContext(this);

    progressDialog = new ProgressDialog(this);

progressDialog.setMessage(getString(org.d3ifcool.finpro.R.string.text_progres
s_dialog));

    button_login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            username = editTextUsername.getText().toString();
            password = editTextPassword.getText().toString();

            if (username.isEmpty()){

editTextUsername.setError(getString(R.string.text_tidak_boleh_kosong));
            } else if (password.isEmpty()){

editTextPassword.setError(getString(R.string.text_tidak_boleh_kosong));
            } else {
                loginPresenter.getLogin(username, password);
            }
        }
    });

    checkUserLogin(sessionManager.getSessionPengguna());
}
```

Dengan Databinding

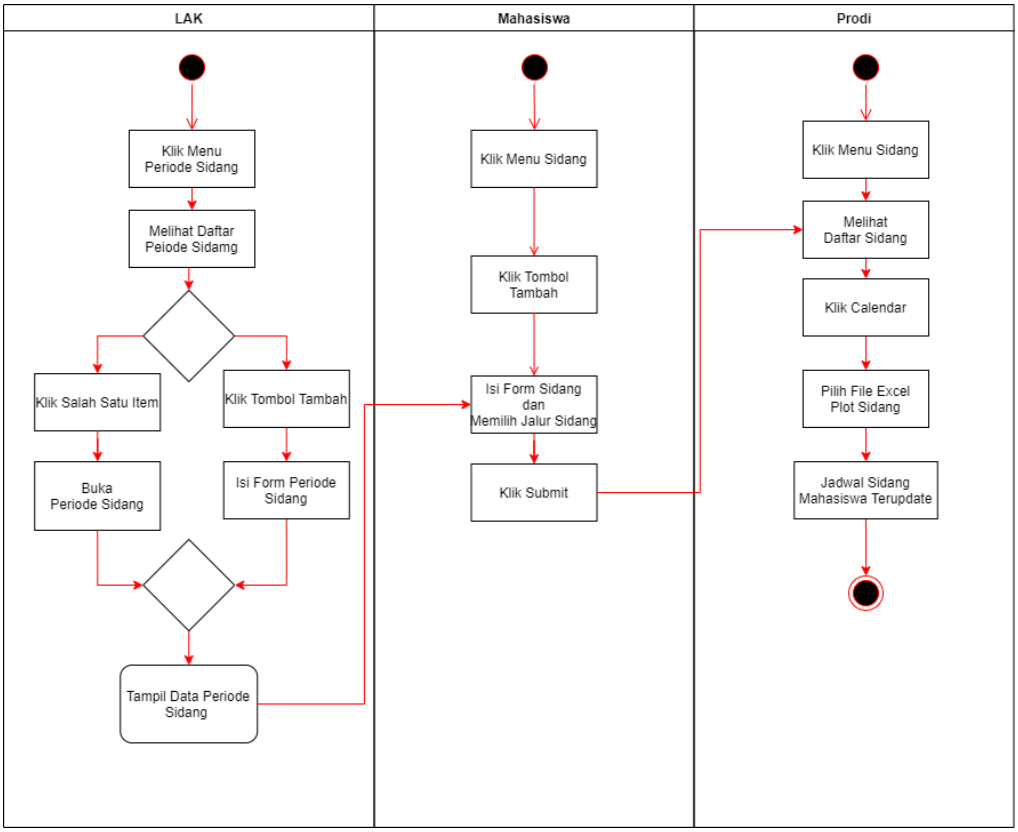
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivitySigninBinding.inflate(layoutInflater)
    setContentView(binding.root)
    supportActionBar?.hide()
    progressDialog = ProgressDialog(this)
    progressDialog.setMessage(getString(R.string.text_progress_dialog))
    user = getSession(USER_SP)
    if (user != null) {
        goToMainPage(user!!.role)
    }
    binding.actMainButtonLogin.setOnClickListener {
        login(
            binding.actMainEdittextUsername.text.toString(),
            binding.actMainEdittextPassword.text.toString()
        )
    }
}
```

Lampiran 7 ActivityDiagram

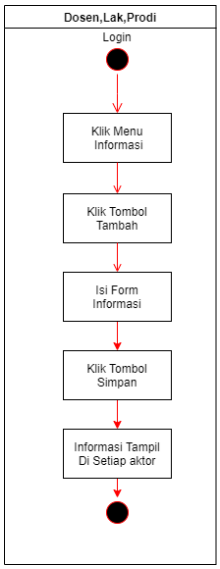
Activity Diagram Bimbingan



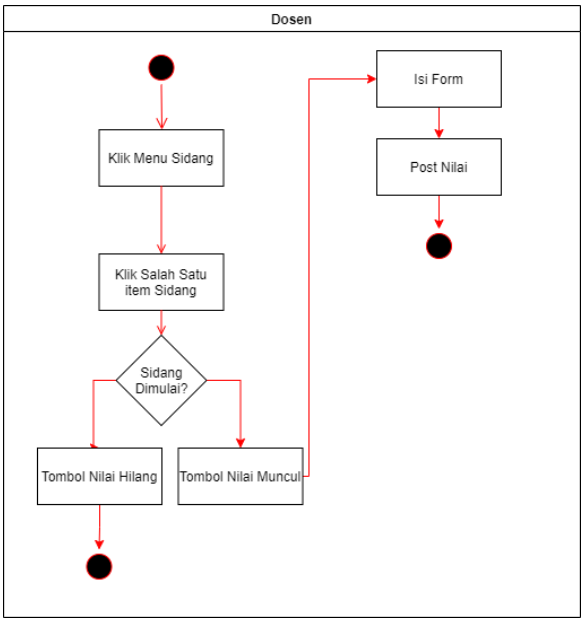
Activity Diagram Sidang



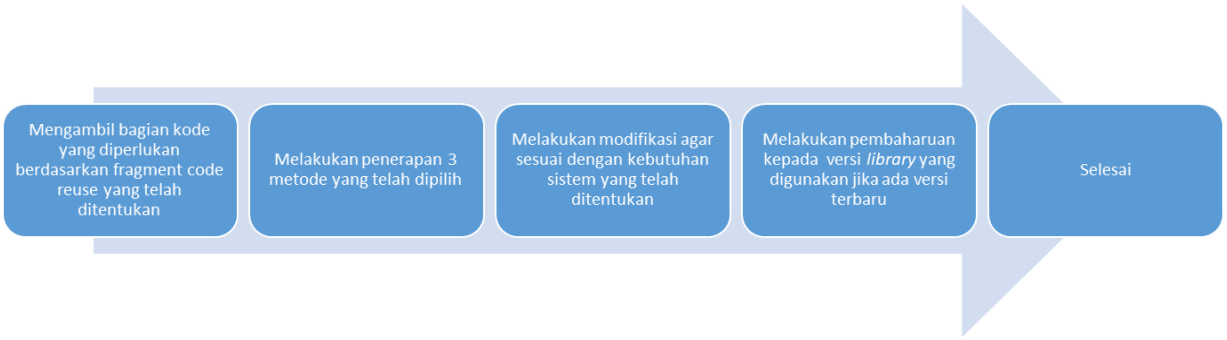
Activity Diagram Informasi



Activity Diagram Informasi



Lampiran 8 Alur Proses Dalam Melakukan Daur Ulang



Lampiran 9 Tabel Kendala Dan Kemudahan Dalam Melakukan Daur Ulang

No	KENDALA	KEMUDAHAN
1	Aplikasi sebelumnya memiliki beberapa library yang *deprecated*	Buku manual dan dokumentasi kode sangat lengkap sehingga mempermudah proses daur ulang
2	Beberapa fitur aplikasi sidang S1 informatika tidak ada pada aplikasi D3 informatika	Tidak perlu membuat semua asset dari awal
3		Lebih fokus untuk melakukan pengembangan pada fitur yang belum ada pada aplikasi sebelumnya