TUGAS AKHIR – KI141502

PENERAPAN POLA PERANCANGAN DAN PENGUKURAN KUALITAS PADA SISTEM INFORMASI AKADEMIK: STUDI KASUS MODUL EKUIVALENSI

Afif Ishamsyah Hantriono
NRP 5113 100 172

Dosen Pembimbing
Dr. Ir. Siti Rochimah, MT.
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

PENERAPAN POLA PERANCANGAN DAN PENGUKURAN KUALITAS PADA SISTEM INFORMASI AKADEMIK: STUDI KASUS MODUL EKUIVALENSI

Afif Ishamsyah Hantriono
NRP 5113 100 172

Dosen Pembimbing
Dr. Ir. Siti Rochimah, MT.
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF DESIGN PATTERN AND QUALITY MEASUREMENT ON ACADEMIC INFORMATION SYSTEM ON CASE STUDY: EQUIVALENCE MODULE

Afif Ishamsyah Hantriono
NRP 5113 100 172

Advisor
Dr. Ir. Siti Rochimah, MT.
Rizky Januar Akbar, S.Kom., M.Eng.

INFORMATICS ENGINEERING DEPARTEMENT
Information Technology Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

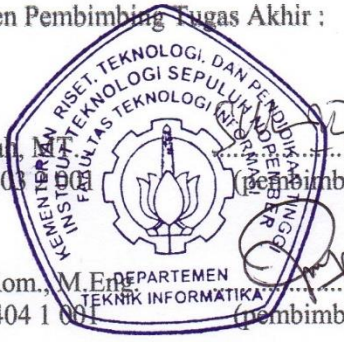
PENERAPAN POLA PERANCANGAN DAN PENGUKURAN KUALITAS PADA SISTEM INFORMASI AKADEMIK: STUDI KASUS MODUL EKUIVALENSI

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
AFIF ISHAMSYAH HANTRIONO
NRP : 5113 100 172

Disetujui oleh Dosen Pembimbing Tugas Akhir :



Dr. Ir. Siti Rochimah, MT. .....
NIP: 19681002 199403 2 001 (pembimbing 1)

Rizky Januar Akbar, S.Kom., M.Eng. .....
NIP: 19870103 201404 1 001 (pembimbing 2)

SURABAYA
JUNI 2017

[Halaman ini sengaja dikosongkan]

PENERAPAN POLA PERANCANGAN DAN PENGUKURAN KUALITAS PADA SISTEM INFORMASI AKADEMIK: STUDI KASUS MODUL EKUIVALENSI

Nama Mahasiswa	: Afif Ishamsyah Hantriono
NRP	: 5113 100 172
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing	: Dr. Ir. Siti Rochimah, MT.
	Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRAK

Sistem informasi akademik adalah sistem yang digunakan suatu institusi pendidikan untuk menjalankan seluruh proses bisnis utama dari kegiatan pendidikan. Dalam penerapan sistem informasi akademik, kualitas perlu diperhatikan agar sistem memiliki kinerja yang baik.

Kualitas perangkat lunak memiliki beberapa standar internasional pengukuran yang akan menentukan sejauh mana sebuah sistem memenuhi kebutuhan yang telah ditentukan oleh semua pemangku kepentingan.

Pada sistem informasi akademik, kualitas dapat dipengaruhi oleh pola perancangan yang diimplementasikan oleh pengembang. Untuk membuktikan pengaruh pola perancangan terhadap kualitas perangkat lunak, maka akan dilakukan perubahan pola perancangan pada sistem informasi akademik. Modul yang dipilih untuk perubahan pola perancangan adalah modul Ekuivalensi.

Setelah perubahan pola perancangan, kualitas modul Ekuivalensi pada sistem informasi akademik akan diukur kembali dan dibandingkan dengan kualitas awal. Hasil yang diharapkan

adalah berubahnya kualitas modul Ekuivalensi pada sistem informasi akademik.

Kata kunci : Ekuivalensi, Kualitas, Pola Perancangan

IMPLEMENTATION OF DESIGN PATTERN AND QUALITY MEASUREMENT ON ACADEMIC INFORMATION SYSTEM ON CASE STUDY: EQUIVALENCE MODULE

Student Name : Afif Ishamsyah Hantriono
NRP : 5113 100 172
Major : Teknik Informatika FTIf-ITS
Advisor : Dr. Ir. Siti Rochimah, MT.
Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRACT

Academic information system is a system used by an educational institution to run all major business processes of educational activities. In the application of Academic Information System, quality needs to be considered for the system to has a good performance.

Software quality has several international measurement standards that will determine the extent to which a system meets the needs set by all stakeholders.

In academic information system, quality can be influenced by the design pattern implemented by the developer. To prove the influence of design pattern to software quality, design pattern will be changed on academic information system. The module chosen for the design pattern change and quality measurement is the Equivalence module.

After changing the design pattern, the quality of Equivalence module on academic information system will be measured again and compared with the initial quality. The expected result is the change of quality of Equivalence module in academic information system.

Keyword: Equivalence, Quality, Design Pattern

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"PENERAPAN POLA PERANCANGAN DAN PENGUKURAN KUALITAS PADA SISTEM INFORMASI AKADEMIK: STUDI KASUS MODUL EKUIVALENSI"**.

Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Teknologi Sepuluh Nopember Surabaya. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Kedua orang tua, dan keluarga penulis, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Direktorat Jenderal Pendidikan Tinggi, terima kasih atas beasiswa bidik misi yang diberikan selama penulis belajar di Teknik Informatika ITS.
4. Ibu Dr. Ir. Siti Rochimah, MT. selaku pembimbing I dan membantu sekaligus membimbing penulis selama pengerjaan Tugas Akhir.
5. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku pembimbing II dan membantu sekaligus membimbing penulis selama pengerjaan Tugas Akhir.
6. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, dan Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA.

7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis belajar di Teknik Informatika ITS.
8. Seluruh staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis belajar di Teknik Informatika ITS.
9. Penulis secara langsung maupun tidak langsung dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2017

Afif Ishamsyah Hantriono

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR LAMPIRAN

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1 Latar Belakang

Sistem informasi akademik adalah sistem yang digunakan suatu institusi pendidikan untuk menjalankan seluruh proses bisnis utama dari kegiatan pendidikan, seperti penilaian dan absensi. Dalam penerapan sistem informasi akademik, kualitas perlu diperhatikan agar sistem memiliki kinerja yang baik.

Kualitas perangkat lunak memiliki beberapa standar internasional pengukuran seperti ISO/IEC 25010 atau ISO/IEC 25023. Setiap standar memiliki parameter kualitas mereka masing-masing. Kualitas sebuah sistem akan menentukan sejauh mana sebuah sistem memenuhi kebutuhan yang telah ditentukan oleh semua pemangku kepentingan, sehingga menghasilkan sebuah nilai.

Pada sistem informasi akademik, kualitas dapat dipengaruhi oleh pola perancangan yang diimplementasikan oleh pengembang. Untuk membuktikan pengaruh pola perancangan terhadap kualitas perangkat lunak, maka akan dilakukan perubahan pola perancangan pada sistem informasi akademik. Modul yang dipilih untuk perubahan pola perancangan dan pengukuran kualitas adalah modul Ekuivalensi. Pola perancangan yang saat ini diimplementasikan pada modul Ekuivalensi akan diubah dengan 2 pola perancangan yaitu Active Record dan Data Mapper.

Setelah implementasi 2 pola perancangan baru, kualitas modul Ekuivalensi pada sistem informasi akademik akan diukur kembali sesuai parameter yang dipilih dan dibandingkan dengan

kualitas yang menggunakan pola perancangan awal. Standar pengukuran yang dipilih adalah ISO 25023 dengan parameter yang dipilih yaitu Usability dan Maintainability.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut.

1. Bagaimana cara menerapkan pola perancangan Active Record dan Data Mapper pada modul Ekuivalensi di sistem informasi akademik?
2. Bagaimana cara mengukur kualitas modul Ekuivalensi di sistem informasi akademik dengan parameter kualitas Usability dan Maintainability sesuai ISO 25023?
3. Bagaimana cara mengevaluasi hasil pengukuran kualitas pada modul Ekuivalensi di sistem informasi akademik?
4. Apakah perubahan pola perancangan akan mempengaruhi kualitas modul Ekuivalensi pada sistem informasi akademik?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, diantaranya adalah sebagai berikut.

1. Standar pengukuran kualitas perangkat lunak yang digunakan adalah ISO/IEC DIS 25023.
2. Sistem informasi akademik ini berbasis *web* dengan bahasa pemrograman Java dengan Framework Spring dan basis data PostgreSQL.
3. Sistem informasi akademik yang digunakan adalah sistem informasi akademik versi penelitian (replika Sistem Informasi Akademik) [1] [2] [3] [4] [5] [6]. Versi ini tidak sama dengan versi yang digunakan pada Institut Teknologi Sepuluh Nopember.
4. Basis data yang digunakan adalah basis data replika dan tidak akan mengalami perubahan.

5. Modul yang diukur kualitasnya dan mengalami perubahan pola perancangan adalah modul Ekuivalensi.

1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah sebagai berikut.

1. Mengetahui cara menerapkan pola perancangan Active Record dan Data Mapper pada modul Ekuivalensi di sistem informasi akademik.
2. Mengetahui cara mengukur kualitas modul Ekuivalensi di sistem informasi akademik dengan parameter kualitas Usability dan Maintainability sesuai ISO 25023.
3. Mengetahui cara mengevaluasi hasil pengukuran kualitas pada modul Ekuivalensi di sistem informasi akademik.
4. Mengetahui apakah perubahan pola perancangan akan mempengaruhi kualitas modul Ekuivalensi pada sistem informasi akademik.

1.5 Manfaat

Manfaat yang diharapkan dari tugas akhir adalah sebagai berikut.

1. Untuk memberikan bukti ilmiah bahwa penerapan pola perancangan dapat mempengaruhi kualitas sistem informasi akademik.
2. Sebagai referensi untuk pengembangan sistem informasi akademik.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir

Proposal Tugas Akhir ini berisi tentang pendahuluan yang meliputi deskripsi latar belakang, rumusan masalah, batasan masalah, tujuan dari pembuatan tugas akhir, dan manfaat dari tugas akhir. Selain itu dijelaskan juga tinjauan pustaka yang digunakan sebagai referensi pembuatan tugas akhir. Pada sub bab metodologi dijelaskan mengenai urutan penyusunan tugas akhir dimulai dari pembuatan proposal sampai pembuatan buku tugas akhir.

2. Studi literatur

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai parameter kualitas perangkat lunak, cara pengukuran kualitas perangkat lunak, ISO 25023, *Design Pattern*, *Enterprise Application Pattern*.

3. Analisis dan desain perangkat lunak

   Analisis dan desain perangkat lunak yang kami kembangkan akan berbasis pada sistem informasi akademik yang telah dikembangkan sebelumnya. Pada tahap ini juga kami melakukan Reverse Engineering terhadap *source code* dari Sistem informasi akademik yang telah dikembangkan sehingga didapatkan diagram paket dan diagram kelas. Kami juga menentukan parameter-parameter uji yang nantinya akan digunakan dalam uji kualitas perangkat lunak.

4. Implementasi perangkat lunak

   Sistem informasi akademik ini dibangun dengan bahasa pemrograman Java dengan Framework Java Spring MVC. Untuk mengembangkan aplikasi ini kami menggunakan Integrated Development Environment (IDE) Eclipse dan PostgreSQL sebagai Relational Database Management System (RDBMS). Tahap implementasi ini dibagi menjadi tiga bagian yaitu uji kualitas tahap pertama, implementasi perubahan pola perancangan, dan uji kualitas tahap kedua.

5. Pengujian dan evaluasi

   Pengujian dilakukan sesuai dengan ketentuan yang telah ada dalam ISO 25023[10] terutama pada penilaian internal

source code. Evaluasi dilakukan dengan cara membandingkan nilai kualitas antara sebelum dan sesudah perubahan pola perancangan.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan teori-teori yang berkaitan dengan rancangan modularitas dan evolusi pada modul Ekuivalensi yang diajukan pada pengimplementasian program.

Bab III Analisis dan Perancangan Sistem

Bab ini akan membahas mengenai alur pengerjaan tugas akhir.

Bab IV Implementasi

Bab ini akan membahas mengenai implementasi perubahan pola perancangan terhadap sistem.

Bab V Pengujian dan Evaluasi

Pada bab ini akan dijelaskan uji kualitas yang dilakukan pada modul Ekuivalensi. Pembahasan pengujian meliputi analisis pengukuran kualitas dan hasil pengukuran sebelum dan sesudah perubahan pola perancangan.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran-saran untuk pengembangan aplikasi pada masa mendatang.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi diagram kelas yang penting pada aplikasi ini.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan rancangan modularitas dan evolusi pada modul Ekuivalensi yang diajukan pada pengimplementasian program.

2.1 Sistem Informasi Akademik

Sistem informasi akademik adalah sistem informasi yang digunakan pada institusi pendidikan. Sistem informasi akademik memiliki beberapa fungsionalitas yang dapat membantu menjalankan proses bisnis dari suatu institusi pendidikan. Pada tugas akhir yang akan kami kembangkan, kami menggunakan Sistem Informasi Akademik versi penelitian (replika Sistem Informasi Akademik).

Pada sistem informasi akademik versi penelitian ini memiliki 6 modul yaitu modul Framework, Domain, Ekuivalensi, Kurikulum, Pembelajaran, dan Penilaian. Pengembangannya sendiri menggunakan bahasa pemrograman Java, Framework Spring, dan basis data PostgreSQL.

2.2 Kualitas Perangkat Lunak

Kualitas perangkat lunak adalah pemenuhan terhadap kebutuhan fungsional dan kinerja yang didokumentasikan secara eksplisit, pengembangan standar yang didokumentasikan secara eksplisit dan sifat-sifat implisit yang diharapkan dari sebuah *software* yang dibangun secara profesional [7]. Sedangkan perangkat lunak sendiri dikatakan berkualitas apabila memenuhi tiga ketentuan pokok sebagai berikut.

- memenuhi kebutuhan pemakai - yang berarti bahwa jika perangkat lunak tidak dapat memenuhi kebutuhan pengguna

perangkat lunak tersebut, maka yang bersangkutan dikatakan tidak atau kurang memiliki kualitas;

- memenuhi standar pengembangan perangkat lunak - yang berarti bahwa jika cara pengambangan perangkat lunak tidak mengikuti metodologi standar, maka hampir dapat dipastikan bahwa kualitas yang baik akan sulit atau tidak tercapai; dan
- memenuhi sejumlah kriteria implisit - yang berarti bahwa jika salah satu kriteria implisit tersebut tidak dapat dipenuhi, maka perangkat lunak bersangkutan tidak dapat dikatakan memiliki kualitas yang baik [8].

2.3 *Software product Quality Requirements and Evaluation (SQuaRE)*

Quality Model adalah dasar dari sistem evaluasi dari kualitas suatu produk. Quality Model menentukan karakteristik kualitas yang akan digunakan dalam mengevaluasi sebuah produk perangkat lunak.

Kualitas sebuah sistem akan menentukan sejauh mana sebuah sistem memenuhi kebutuhan yang telah ditentukan oleh semua pemangku kepentingan, sehingga menghasilkan sebuah nilai. Semua kebutuhan dari para pemangku kepentingan akan direpresentasikan di dalam Quality Model, yang akan mengategorikan kualitas produk menjadi karakteristik dan sub-karakteristik [9].

Quality Model sendiri memiliki beberapa standar internasional yaitu ISO/IEC 25010 atau ISO/IEC 25023 [10]. Parameter kualitas yang dipilih pada tugas akhir ini adalah sesuai dengan ISO/IEC 25023 dijelaskan pada tabel berikut.

Tabel 2.1 Parameter kualitas berdasarkan ISO 25023

No	Kualitas	Sub-kualitas
1	Functional Suitability	Functional Completeness
		Functional Correctness
		Functional Appropriateness

No	Kualitas	Sub-kualitas
2	Performance	Time Behaviour
		Resource Utilization
		Capacity
3	Compatibility	Co-existence
		Interoperability
4	Usability	Appropriateness Recognisability
		Learnability
		Operability
		User Error Protection
		User Interface Aesthetics
		Accessibility
5	Reliability	Maturity
		Availability
		Fault Tolerance
		Recoverability
6	Security	Confidentiality
		Integrity
		Non-repudiation
		Accountability
		Authenticity
7	Maintainability	Modularity
		Reusability
		Analysability
		Modifiability
		Testability
8	Portability	Adaptability
		Installability
		Replaceability

Penjelasan parameter kualitas pada Tabel 2.1 adalah sebagai berikut:

1. Functional Suitability

   Functional Suitability menilai sejauh mana produk atau sistem menyediakan fitur-fitur yang memenuhi kebutuhan dalam kondisi tertentu. Functional Suitability memiliki 3 sub-kualitas yaitu:

a) Functional Completeness yang mengukur sejauh mana sekumpulan fungsi dapat memenuhi sebuah tugas atau kebutuhan pengguna. Functional Completeness memiliki sebuah poin kualitas yaitu Functional Coverage, yang menghitung proporsi fungsi yang diimplementasi.
b) Functional Correctness yang mengukur sejauh mana sebuah sistem menyediakan hasil yang benar sesuai kebutuhan. Functional Correctness memiliki sebuah poin kualitas dengan nama yang sama yaitu Functional Correctness, yang menghitung proporsi fungsi yang mempunyai hasil yang benar.
c) Functional Appropriateness yang mengukur sejauh mana fungsi dapat memfasilitasi pencapaian suatu kebutuhan. Functional Appropriateness memiliki sebuah poin kualitas dengan nama yang sama yaitu Functional Appropriateness, yang menghitung proporsi fungsi yang dibutuhkan pengguna untuk memenuhi suatu kebutuhan.

2. Performance

Performance menilai sejauh mana performa relatif terhadap jumlah sumber daya yang digunakan dalam kondisi tertentu. Performance memiliki 5 sub-kualitas yaitu:

a) Time Behaviour yang mengukur sejauh mana respon dan waktu ekseskusi proses dapat memenuhi suatu kebutuhan. Time Behaviour memiliki 2 poin kualitas yaitu:
    - Mean Response Time yang menghitung waktu rata-rata dari sistem untuk menjalankan suatu aksi pengguna.
    - Response Time Conformance yang mengukur seberapa baik waktu respon dalam memenuhi suatu target.

- Mean Turnaround Time yang mengukur rata-rata waktu yang dibutuhkan untuk menyelesaikan sebuah proses yang bersifat asynchronous.
- Turnaround Time Conformance yang menghitung waktu *turnaround* untuk memenuhi sebuah target.
- Throughput Conformance yang mengukur apakah *throughput* memenuhi target.

b) Resource Utilization yang mengukur sejauh mana jumlah dan tipe sumber daya yang digunakan sistem ketika memenuhi suatu kebutuhan. Resource Utilization mempunyai 5 poin kualitas yaitu:
   - Mean Processor Utilization yang mengukur waktu *processor* untuk memenuhi sekumpulan tujuan.
   - Mean Memory Utilization yang mengukur jumlah memori yang dipakai untuk menyelesaikan suatu tujuan.
   - Mean I/O Devices Utilization yang mengukur waktu sibuk Device I/O dalam menjalankan suatu tujuan.
   - Storage Utilization yang mengukur ketersediaan *storage* dalam menjalankan suatu tujuan.
   - Bandwith Utilization yang mengukur seberapa *bandwith* yang terpakai untuk menjalankan suatu tujuan.

c) Capacity yang mengukur sejauh mana batas maksimal suatu produk atau sistem memenuhi kebutuhan. Capacity memiliki 3 poin kualitas yaitu:
   - Transaction Processing Capacity Conformance yang menghitung seberapa banyak transaksi konkuren dapat diproses

pada waktu tertentu terhadap target yang ditentukan.

- User Access Capacity Conformance yang mengukur seberapa banyak pengguna dapat mengakses sistem secara bersamaan pada waktu tertentu.
- User Access Increase Conformance yang mengukur seberapa banyak pengguna yang dapat ditambahkan sesuai dengan kebutuhan sistem.

3. Compatibility

Compatibility menilai sejauh mana suatu produk, sistem, atau komponen dapat bertukar informasi ke produk, sistem, atau komponen lain dan menjalankan fungsi yang dibutuhkan dalam lingkup perangkat keras atau perangkat lunak yang sama. Compatibility memiliki 2 sub-kualitas yaitu:

a) Co-existence yang mengukur sejauh mana sebuah produk dapat menjalankan fungsi secara efisien sembari membagi lingkungan dan sumber daya dengan produk lain tanpa adanya konflik dengan produk tersebut. Co-existence memiliki sebuah poin kualitas dengan nama yang sama yaitu Co-existence, yang mengukur proporsi produk tertentu dapat berbagi lingkungan dengan produk perangkat lunak lain tanpa terjadi konflik.
b) Interoperability yang mengukur sejauh mana dua atau lebih sistem dapat saling bertukar informasi dan menggunakan informasi tersebut. Interoperability memiliki 3 poin kualitas yaitu:
    - Data Exchange Format Conformance yang mengukur proporsi format data tertentu apakah dapat ditukarkan dengan perangkat lunak atau sistem lain.

- Data Exchange Protocol Conformance yang mengukur proporsi protokol pertukaran data yang ditetapkan.
- External Interface Conformance yang mengukur proporsi External Interface (antarmuka dengan perangkat lunak dan sistem lainnya) yang telah dilaksanakan.

4. Usability

Usability menilai sejauh mana suatu produk atau sistem dapat digunakan oleh pengguna yang dituju untuk mencapai suatu tujuan dengan efisien. Usability memiliki 6 sub-kualitas yaitu:

a) Appropriateness Recognisability yang mengukur sejauh mana pengguna dapat mengenali apakah produk atau sistem dapat memenuhi kebutuhan. Appropriateness Recognisability memiliki 3 poin kualitas yaitu:
   - Description Completeness yang mengukur proporsi dari skenario yang dideskripsikan pada deskripsi produk atau dokumen pengguna.
   - Demonstration Capability yang mengukur proporsi dari fungsi yang mampu mendemonstrasikan kepada pengguna untuk menilai kelayakan.
   - Entry Point Self-descriptiveness yang mengukur proporsi dari *landing pages* dari *website* yang menjelaskan tujuan dari *website*.

b) Learnability yang megukur sejauh mana sebuah produk atau sistem dapat digunakan oleh pengguna untuk mencapai suatu tujuan dengan mempelajari produk tersebut dengan efektif dan efisien. Learnability memiliki 4 poin kualitas yaitu:
   - User Guidance Completeness yang mengukur proporsi petunjuk yang menjelaskan secara

detil kepada pengguna tentang cara penggunaan sistem.

- Entry Fields Defaults yang mengukur entri yang memiliki nilai *default*.
- Error Messages Understandability yang mengukur proporsi kesalahan pesan beserta alasan dan solusinya.
- Self-explanatory User Interface yang mengukur proporsi tampilan yang langsung dimengerti pengguna tanpa harus dipelajari sebelumnya.

c) Operability yang mengukur sejauh mana produk atau sistem memiliki atribut yang memudahkan penggunaan. Operability memiliki 8 poin kualitas yaitu:

- Operational Consistency yang mengukur proporsi fungsi yang memiliki tampilan dan aturan yang konsisten dalam tugasnya.
- Message Clarity yang mengukur proporsi pesan dari sistem yang dapat dimengerti.
- Functional Customizability yang mengukur proporsi fitur dan prosedur yang bisa bebas diatur oleh pengguna.
- User Interface Customizability yang mengukur Proporsi elemen *user interface* yang tampilannya bisa diatur.
- Monitoring Capability yang mengukur proporsi *state* dari fitur yang bisa dimonitor saat dijalankan.
- Undo Capability yang mengukur proporsi dari fitur yang mempunyai opsi konfirmasi ulang atau *undo*.
- Terminology Understandability yang mengukur proporsi istilah-istilah dalam *user interface* yang dipahami oleh pengguna.

- Appearance Consistency yang mengukur proporsi dimana *user interface* dengan isi sama mempunyai tampilan yang sama.

d) User Error Protection yang mengukur sejauh mana sebuah sistem dapat melindungi pengguna dari *error*. User Error Protection memiliki 3 poin kualitas yaitu:
   - Avoidance of User Operation Error yang mengukur proporsi pencegahan tindakan pengguna yang dapat menyebabkan *error*.
   - User Entry Error Correction yang mengukur proporsi sugesti perbaikan kepada pengguna ketika terjadi kesalahan.
   - User Error Recoverability yang mengukur proporsi *error* yang dapat diperbaiki oleh sistem.

e) User Interface Aesthetics yang mengukur sejauh mana User Interface dapat menyediakan interaksi yang memuaskan pengguna. User Interface Aesthetics memiliki sebuah poin kualitas yaitu Appeareance Aesthetics of User Interface yang menghitung proporsi User Interface yang memiliki desain dan tampilan memuaskan.

f) Accessibility yang mengukur sejauh mana produk atau sistem dapat digunakan oleh berbagai macam karakteristik dan kapabilitas pengguna untuk memenuhi suatu kebutuhan. Accessibility memiliki 5 poin kualitas yaitu:
   - Accessibility for Users with Cognitive Disability yang mengukur sejauh mana pengguna potensial dengan kemampuan kognitif yang terbatas berhasil menggunakan sistem.
   - Accessibility for Users with Physical Disability yang mengukur sejauh mana

pengguna potensial dengan kemampuan fisik yang terbatas berhasil menggunakan sistem.

- Accessibility for Users with Hearing Disability yang mengukur sejauh mana pengguna potensial dengan kemampuan pendengaran yang terbatas berhasil menggunakan sistem.
- Accessibility for Users with Visual Disability yang mengukur sejauh mana pengguna potensial dengan kemampuan pengelihatan yang terbatas berhasil menggunakan sistem.
- Supported Languages mengukur proporsi bahasa yang dibutuhkan.

5. Reliability

Reliability menilai sejauh mana suatu sistem, produk, atau komponen dapat mengerjakan sebuah fungsi dalam kondisi tertentu dan dalam waktu tertentu. Reliability memiliki 4 sub-kualitas yaitu:

a) Maturity yang mengukur sejauh mana kehandalan sebuah sistem dalam memenuhi kebutuhan secara normal. Maturity mempunyai 4 poin kualitas yaitu:
   - Fault Correction yang mengukur berapa proporsi terdeteksi kesalahan terkait Reliability yang telah diperbaiki.
   - Mean Time Between Failure Conformance (MBTF) yang mengukur rata rata MBTF saat sistem dijalankan dibandingkan dengan *threshold*.
   - Failure Rate Conformance yang mengukur berapa banyak kegagalan yang terdeteksi selama periode yang didefinisikan dibandingkan dengan *threshold* yang ditetapkan untuk tingkat kegagalan.
   - Test Coverage yang mengukur jumlah tes yang diperlukan apakah telah dilakukan.

b) Availability yang mengukur sejauh mana produk atau sistem dapat beroperasi dan dapat diakses ketika dibutuhkan. Availability memiliki 3 poin kualitas yaitu:
    - Availability of System Operation yang mengukur berapa proporsi waktu operasi sistem yang sebenarnya disediakan.
    - Mean Down Time Conformance yang mengukur seberapa sistem tidak bisa dijalankan dibandingkan dengan *threshold* yang ditentukan.
    - System Availability in Special Days yang mengukur berapa lama sistem beroperasi selama hari-hari khusus.
c) Fault Tolerance yang mengukur sejauh mana sebuah sistem beroperasi ketika ada kesalahan pada perangkat lunak atau perangkat keras. Fault Tolerance memiliki 3 poin kualitas yaitu:
    - Failure Avoidance yang mengukur berapa proporsi pola kesalahan yang telah dikendalikan untuk menghindari kegagalan kritis dan serius.
    - Redundancy of Components yang mengukur berapa proporsi komponen sistem yang dimasukan secara berlebihan untuk menghindari kegagalan sistem.
    - Mean Fault Notification Time Conformance yang mengukur seberapa cepat sistem melaporkan terjadinya kesalahan.
d) Recoverability yang mengukur sejauh mana sebuah sistem dapat mengembalikan data dan keadaan ke kondisi awal ketika terjadi sebuah kegagalan sistem. Recoverability memiliki 2 poin kualitas yaitu:
    - Mean Recovery Time Conformance yang mengukur seberapa baik waktu sistem untuk

memulihkan diri dari kegagalan sistem dibandingkan dengan batas yang telah ditentukan.

- Backup Data Completeness yang mengukur proporsi data pada sistem dilakukan *backup* secara berkala.

6. Security

Security menilai sejauh mana suatu produk atau sistem dapat melindungi informasi dan data sehingga seseorang, sistem, atau produk lain dapat mengakses data atau informasi tersebut sesuai dengan hak akses mereka masing-masing. Security memiliki 5 sub-kualitas yaitu:

a) Confidentiality yang mengukur sejauh mana suatu data pada sistem atau produk hanya dapat diakses oleh pengguna yang memiliki hak akses. Confidentiality memiliki 3 poin kualitas yaitu:
   - Access Controllability yang mengukur proporsi data rahasia yang dilindungi dari akses tanpa izin.
   - Data Encryption Correctness yang mengukur proporsi penerapan enkripsi/dekripsi apakah sudah sesuai dengan spesifikasi.
   - Strength of Cryptographic Algorithm yang mengukur seberapa banyak algoritma kriptografi yang digunakan.

b) Integrity yang mengukur sejauh mana sebuah sistem mencegah akses tak dikenal yang dapat memodifikasi sistem tersebut. Integrity memiliki 3 poin kualitas yaitu:
   - Data Integrity Conformance yang mengukur sampai sejauh mana kerusakan data atau modifikasi terjaga dari akses yang tidak sah.
   - Internal Data Corruption Prevention yang mengukur sejauh mana metode pencegahan

yang tersedia untuk kerusakan data diimplementasikan.

- Validity of Array Accesses yang mengukur sejauh mana input untuk askes pengguna divalidasi.

c) Non-repudiation yang mengukur sejauh mana setiap aksi atau peristiwa yang terjadi pada sistem dapat dibuktikan ketika dibutuhkan. Non-repudiaton memiliki sebuah poin kualitas yaitu Utilization of Digital Signature yang mengukur proporsi kejadian yang diproses dengan *digital signature*.

d) Accountability yang mengukur sejauh mana aksi dari sebuah komponen dapat dilacak secara unik ke komponen tersebut. Accountability memiliki 2 poin kualitas yaitu:

- Access Auditability yang mengukur seberapa lengkap adalah jejak audit dari akses pengguna ke sistem atau data.
- System Log Retention Conformance yang mengukur seberapa persen waktu yang dibutuhkan agar log sistem dapat disimpan dengan stabil.

e) Authenticity yang mengukur sejauh mana identitas suatu subjek atau sumber daya dapat dibuktikan oleh orang yang melakukan klaim terhadap identitas tersebut. Authenticity memiliki 2 poin kualitas yaitu:

- Authentication Protocol Conformance yang mengukur seberapa baik autentikasi sistem terhadap identitas suatu subjek atau sumber daya.
- Authentication Rules Conformance yang mengukur proporsi aturan autentikasi yang diperlukan untuk menetapkan bahwa sistem dalam kondisi aman.

7. Maintainability

Maintainability menilai sejauh mana seberapa efektif dan efisien suatu produk atau sistem dapat dimodifikasi oleh pengelola. Maintainability memiliki 5 sub-kualitas yaitu:

a) Modularity yang mengukur sejauh mana perubahan sebuah komponen pada sistem hanya dapat berpengaruh kecil kepada komponen lain. Modularity mempunyai 2 poin kualitas yaitu:
   - Coupling of Components Conformance yang mengukur seberapa independen sebuah komponen dan berapa banyak komponen yang bebas dari efek perubahan komponen lain dalam sistem.
   - Cyclomatic Complexity yang mengukur jumlah modul yang mempunyai Cyclomatic Complexity dengan jumlah yang dapat diterima.

b) Reusability yang mengukur sejauh mana sebuah aset dapat digunakan lebih dari satu sistem. Reusablity memiliki 2 poin kualitas yaitu:
   - Reusability of Assets yang mengukur jumlah aset yang bisa digunakan ulang.
   - Conformance to Coding Rules yang mengukur jumlah modul yang dikembangkan sesuai *coding rules*.

c) Analysability yang mengukur apakah memungkinkan untuk menilai suatu dampak perubahan pada produk atau sistem, atau untuk mendiagnosis kekurangan dan kegagalan produk yang dapat terjadi, atau untuk mengidentifikasi bagian yang akan dimodifikasi. Analysability memiliki 3 poin kualitas yaitu:
   - System Log Completeness Conformance yang mengukur seberapa jauh jangkauan log sistem dalam mencatat operasi-operasi dalam sistem.

- Diagnosis Function Effectiveness yang mengukur proporsi sejauh mana implementasi fitur yang sesuai dengan kebutuhan dibandingkan dengan fitur yang telah diimplementasikan.
- Diagnosis Function Sufficiency Conformance yang mengukur sejauh mana fitur-fitur memenuhi spesifikasi kebutuhan.

d) Modifiability yang mengukur sejauh mana modifikasi dapat dilakukan dengan efektif dan efisien tanpa mengurangi kualitas sistem. Modifiability memiliki 3 poin kualitas yaitu:
   - Modification Efficiency yang mengukur efisiensi waktu yang dihabiskan untuk membuat modifikasi dibandingkan dengan waktu yang diharapkan.
   - Modification Correctness yang mengukur proporsi modifikasi yang telah diterapkan dengan benar.
   - Modification Capability yang mengukur seberapa banyak waktu yang digunakan untuk memodifikasi sesuatu yang diperlukan dalam kurun waktu tertentu.

e) Testability yang mengukur sejauh mana kriteria sebuah tes dapat diterapkan ke sebuah sistem dan apakah beberapa tes bisa dilakukan untuk memenuhi kriteria-kriteria tersebut. Testability memiliki 3 poin kualitas yaitu:
   - Test Function Completeness Conformance yang mengukur seberapa lengkap uji coba terhadap sistem.
   - Autonomous Testability yang mengukur seberapa kemampuan uji coba secara *autonomous*.

- Test Restartability yang mengukur seberapa mudah uji coba ulang setelah melakukan perbaikan sistem.

8. Portability

Portability menilai sseberapa efektif dan efisien suatu sistem, produk, atau komponen dapat dipindahkan dari suatu lingkungan operasional ke lingkungan operasional lain. Portability memiliki 3 sub-kualitas yaitu:

a) Adaptability yang mengukur sejauh mana sebuah sistem dapat menyesuaikan diri dengan perubahan perangkat keras atau perangkat lunak secara efektif dan efisien. Adaptability memiliki 3 poin kualitas yaitu:
   - Hardware Enviromental Adaptability yang mengukur apakah sistem perangkat lunak dapat beradaptasi dengan perangkat-perangkat keras yang berbeda.
   - System Software Enviromental Adaptability yang mengukur apakah sistem perangkat lunak dapat beradaptasi dengan lingkungan perangkat lunak yang berbeda.
   - Operational Enviromental Adaptability yang mengukur apakah sistem perangkat lunak dapat beradaptasi dengan lingkungan operasional yang berbeda.

b) Installability yang mengukur sejauh mana sebuah sistem dapat ditempatkan atau dikeluarkan dari lingkungan tertentu. Installability memiliki 2 poin kualitas yaitu:
   - Installation Time Efficiency yang mengukur seberapa efisien waktu yang diperlukan untuk instalasi dibanding waktu yang diharapkan.
   - Ease of Installation yang mengukur seberapa mudah pengguna melakukan instalasi sebuah sistem ke suatu lingkungan operasional.

c) Replaceability yang mengukur sejauh mana sebuah produk dapat tergantikan oleh produk lain yang sejenis. Replaceability memiliki 4 poin kualitas yaitu:
   - Usage Similarity yang mengukur proporsi fungsi pengguna pada program pengganti yang dapat dikerjakan dengan mudah tanpa pembelajaran lebih lanjut.
   - Product Quality Equivalence yang mengukur proporsi pengukuran kualitas yang terpenuhi setelah mengganti program sebelumnya dengan program yang sekarang.
   - Functional Inclusiveness yang mengukur apakah fungsi yang mirip dapat dikerjakan setelah mengganti program sebelumnya dengan program yang sekarang.
   - Data Reusability / Import Capability yang mengukur apakah data yang sama dapat digunakan setelah mengganti program sebelumnya dengan program yang sekarang.

2.4 Uji Coba Perangkat Lunak

Menurut IEEE Std 610.12 uji coba perangkat lunak memiliki dua pengertian, pengertian pertama adalah proses penggunaan sistem atau komponen dalam kondisi yang tertentu dengan pengamatan dan pencatatan hasil serta membuat evaluasi dari aspek sistem atau komponen. Sedangkan pengertian kedua adalah proses menganalisis perangkat lunak untuk memperoleh perbedaan antara hasil sistem dan kebutuhan serta melakukan evaluasi terhadap fitur-fiturnya [11].

Berdasarkan konsepnya, uji coba perangkat lunak dibagi menjadi dua jenis yaitu

1. Black-box *testing*. Uji coba ini berfokus pada fungsionalitas dari suatu perangkat lunak baik secara fungsi maupun hasil tanpa memperhatikan proses di dalam sistem.

2. White-box *testing*. Uji coba ini adalah uji coba terhadap proses dalam sistem. Uji coba ini dapat menghasilkan detail mekanisme dan proses di dalam suatu perangkat lunak.

2.5 Pola Perancangan Aplikasi *Enterprise*

Pola perancangan aplikasi *enterprise* merupakan pola-pola yang sering digunakan untuk mengembangkan aplikasi *enterprise*. Pada bukunya "Pattern of Enterprise Application Architecture", Martin Fowler memberikan penjelasan tentang bagian-bagian pola perancangan yang dapat digunakan untuk mengembangkan aplikasi *enterprise*.

Tabel 2.2 Pola perancangan aplikasi *enterprise*

No	Tipe	Pola Perancangan
1	Domain Logic Pattern	Transaction Script
		Domain Model
		Table Module
		Service Layer
2	Data Source Architecture Pattern	Table Data Gateway
		Row Data Gateway
		Active Record
		Data Mapper
3	Object Relational Behavioral	Unit of Work
		Identity Map
		Lazy Load
4	Object Relational Structural Pattern	Identity Field
		Foreign Key Mapping
		Association Table Mapping
		Dependent Mapping
		Embedded Value
		Serialized LOB
		Single Table Inheritance
		Class Table Inheritance
		Concrete Table Inheritance
		Inheritance Mappers

No	Tipe	Pola Perancangan
5	Object- Relational Metadata Mapping Patterns	Metadata Mapping
		Query Object
		Repository
6	Web Presentation Patterns	Model View Controller
		Page Controller
		Front Controller
		Template View
		Transform View
		Two Step View
		Application Controller
7	Distribution Patterns	Remote Facade
		Data Transfer Object
8	Offline Concurrency Patterns	Optimistic Offline Lock
		Pessimistic Offline Lock
		Coarse-Grained Lock
		Implicit Lock
9	Session State Patterns	Client Session State
		Server Session State
		Database Session State

Penjelasan tujuan dari tipe pola perancangan pada Tabel 2.2 adalah sebagai berikut:

1. Domain Logic mempunyai tujuan untuk mengelola model bisnis dari suatu sistem. Ada 4 pola perancangan pada Domain Logic yaitu:
   a) Transaction Script yang mengatur *business logic* dimana satu prosedur hanya mengatur satu *request*.
   b) Domain Model yaitu sebuah model objek berisi data dan tingkah lakunya.
   c) Table Module yaitu sebuah *instance* yang mengatur *business logic* untuk seluruh baris pada tabel basis data.
   d) Service Layer yang mengatur batas aplikasi dengan lapisan-lapisan *service* yang melakukan berbagai

operasi dan mengatur respon dari setiap operasi tersebut.

2. Data Source Architecture bertujuan untuk membentuk hubungan antara Domain Logic dengan basis data. Ada 4 pola perancangan pada Data Source Architecture yaitu:
   a) Table Data Gateway yaitu sebuah objek yang berguna sebagai gerbang dari setiap tabel pada basis data. Satu objek mengatur seluruh baris pada tabel tersebut.
   b) Row Data Gateaway yaitu sebuah objek yang mewakili sebuah baris pada tabel dari basis data.
   c) Active Record adalah kelas yang isinya adalah data dan tingkah laku dari data-data tersebut.
   d) Data Mapper memisahkan antara *domain* dan *data source* sehingga perpindahan data antara objek dan basis data menjadi independen.
3. Object Relational Behavioral bertujuan untuk mengatur pola pemanggilan dan penyimpanan suatu objek ke basis data. Ada 3 pola perancangan pada Object Relational Behavioral yaitu:
   a) Unit of Work yang mencatat semua yang terjadi saat transaksi bisnis dan menyediakan daftar solusi operasi yang dibutuhkan untuk mengubah basis data.
   b) Identity Map yang memastikan setiap objek hanya dipanggil satu kali dengan memetakan setiap pemanggilan.
   c) Lazy Load yaitu sebuah objek yang tidak berisi data yang dibutuhkan namun mengetahui bagaimana cara memanggil data tersebut.
4. Object Relational Structural bertujuan untuk mengorganisir pemetaan antara objek yang berada pada program/memori dengan basis data serta mengorganisir struktur penurunan kelas objek dan pemetaannya ke basis data. Ada 10 pola perancanga pada Object Relational Structural yaitu:

a) Identity Field yang menyimpan ID dari basis data ke dalam sebuah objek.
b) Foreign Key Mapping yang memetakan asosiasi antara objek dengan *foreign key* antar tabel.
c) Association Table Mapping yang menyimpan asosiasi sebagai tabel dengan *foreign key* ke tabel yang dihubungkan dengan asosiasi.
d) Dependent Mapping yang memiliki satu *class* yang berfungsi untuk melakukan pemetaan untuk *child class*.
e) Embedded Value yang memetakan objek ke beberapa baris pada tabel objek lain.
f) Serialized LOB yang menyimpan graf dari beberapa objek ke satu objek besar yang disimpan di basis data.
g) Single Table Inheritance yang merepresentasikan hierarki pewarisan dari *class* ke satu tabel.
h) Class Table Inheritance yang merepresentasikan hierarki pewarisan ke satu tabel untuk setiap *class*.
i) Concrete Table Inheritance yang merepresentasikan hierarki pewarisan dengan satu table per *concrete class* pada hierarki.
j) Inheritance Mappers yaitu struktur organisasi pemetaan basis data yang mengatur hierarki pewarisan.

5. Object Relational Metadata Mapping bertujuan untuk mengorganisir pemetaan dalam bentuk metadata. Ada 3 pola perancangan pada Object Relational Metadata Mapping yaitu:
   a) Metadata Mapping yang memegang detail dari pemetaan objek metadata.
   b) Query Object yaitu sebuah objek yang merepresentasikan *query* pada basis data.
   c) Repository yang memperantarai *domain* dan pemetaan data dengan collection untuk mengakses objek *domain*.

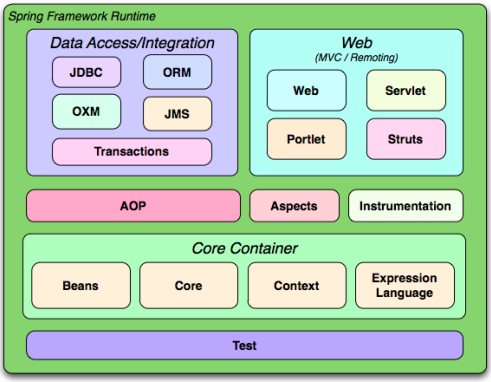
6. Web Presentation bertujuan untuk mengorganisir pemanggilan dari tampilan sebuah program. Web Presentation mempunyai 7 pola perancangan yaitu:
   a) Model View Controller yang membagi peran antara User Interface dan pengaturan data.
   b) Page Controller yaitu sebuah objek yang mengendalikan *request* dari salah satu halaman dalam *website*.
   c) Front Controller yaitu sebuah objek yang mengendalikan seluruh halaman pada *website*.
   d) Template View yang menerjemahkan informasi dalam bentuk HTML.
   e) Transform View yaitu sebuah *view* memproses setiap elemen data *domain* dan mengubahnya ke HTML.
   f) Two-Step View yang membentuk *domain* data menjadi sebuah *logical page*, lalu mengubahnya ke HTML.
   g) Application Controller yaitu sebuah titik pusat yang mengatur navigasi layar dan jalannya aplikasi.
7. Distribution bertujuan untuk mengorganisir proses distribusi objek. Distribution mempunyai 2 pola perancangan yaitu:
   a) Remote Façade yang mengombinasikan beberapa metode yang mirip untuk mengurangi *latency* dan *network traffic*.
   b) Data Transfer Object yaitu sebuah objek yang membawa data antar proses untuk mengurangi pemanggilan *method*.
8. Offline Concurrency bertujuan untuk mengorganisir beberapa proses yang berjalan bersama-sama dengan menggunakan sumber daya yang sama. Offline Concurrency mempunyai 4 pola perancangan yaitu:
   a) Optimistic Offline Lock yang mencegah konflik antar transaksi bisnis dengan mendeteksi konflik lalu memutar kembali transaksi.

b) Pessimistic Offline Lock yang mencegah konflik antar transaksi dengan hanya membiarkan satu transaksi yang berjalan.
c) Coarse-Grained Lock yang mengunci objek yang berhubungan dengan satu *lock*.
d) Implicit Lock yang membiarkan *framework* atau *layer supertype* untuk mendapatkan *offline lock*.

9. Session State bertujuan untuk mengelola penempatan dan penggunaan *session*. Session State memiliki 3 pola perancangan yaitu:
   a) Client Session State yang menyimpan *session* dalam *client*.
   b) Server Session State yang menyimpan *session* dalam *server*.
   c) Database Session State yang menyimpan *session* dalam basis data.

2.6 Kerangka Kerja Spring

Spring [6] adalah salah satu kerangka kerja aplikasi untuk aplikasi berbasis Java atau sering disebut dengan JEE (Java Enterprise Edition). Spring menangani infrastruktur sistem, sehingga kita dapat berfokus pada lapisan aplikasi. Kerangka kerja Spring terdiri dari fitur-fitur yang diatur hingga 20 modul. Modul-modul ini dikelompokkan menjadi Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation dan Test.

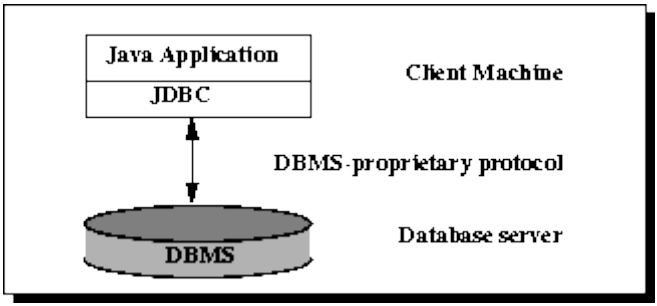


Gambar 2.1 Gambaran umum kerangka kerja Spring

Dalam pengerjaan tugas akhir ini, penggunaan modul kerangka kerja Spring yang ditunjukkan pada Gambar 2.1 adalah JDBC dan Transaction pada akses data/integrasi, Web-Servlet, Beans dan Context pada Core Container.

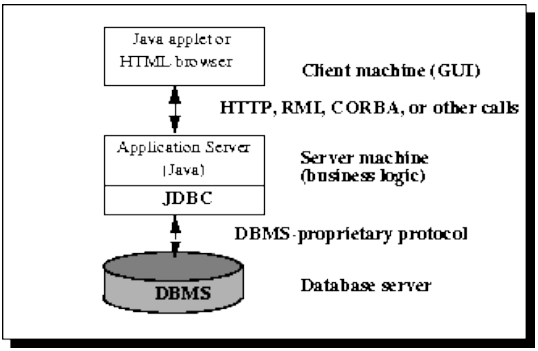
2.6.1 JDBC (Java Database Connectivity)

JDBC API merupakan Java API yang mampu mengakses beragam jenis data tabular, terutama data yang tersimpan dalam basis data relasional. Kemampuan JDBC terdiri dari 3 hal, yaitu menyambungkan sumber data dari basis data, mengirimkan *query* dan perubahan pada basis data, melakukan pengolahan dan temu kembali data yang diterima dari basis data untuk *query* yang dikirimkan pengguna. Arsitektur JDBC mendukung proses model untuk arsitektur dua lapisan (*two-tier*) dan tiga lapisan (*three-tier*) untuk akses ke basis data. Pada Gambar 2.2 akan dijelaskan mengenai visualisasi arsitektur dari JDBC.



Gambar 2.2 Arsitektur JDBC pada arsitektur *Two-Tier*

Pada Gambar 2.2 aplikasi Java atau biasa disebut dengan Java Applet melakukan penyampaian data langsung pada sumber data. Tentunya ini dibutuhkan JDBC yang dapat mengakses kebutuhan aplikasi terhadap data di sumber data. Pengguna mengirimkan *query* pada basis data atau sumber data lainnya dan hasil data akan dikirimkan kembali pada pengguna. Sumber data pada arsitektur ini ditempatkan pada mesin lainnya yang terkoneksi dengan jaringan internet.



Gambar 2.3 Arsitektur JDBC pada arsitektur *Three-Tier*

Berbeda dengan arsitektur *three-tier* pada Gambar 2.3, perintah-perintah akan dikirimkan pada *"middle tier"* dari *service*, kemudian perintah-perintah tersebut diteruskan menuju sumber

data. Sumber data akan melakukan pengolahan perintah dan mengirimkan permintaan data kembali pada *"middle tier"*dan dikirimkan ke pengguna.

2.6.2 Spring-Transaction Manager

Sebuah transaksi basis data terdiri dari urutan aksi yang dianggap sebagai sebuah satuan pekerjaan dalam Spring. Aksi-aksi tersebut harus dapat berjalan seluruhnya atau tidak. Manajemen transaksi menjadi hal terpenting untuk aplikasi *enterprise* berorientasi RDBMS agar data dapat dijamin kebenarannya dan konsisten.

Konsep dari transaksi dijelaskan dengan 4 komponen yang sering disingkat menjadi ACID (Atomicity, Consistency, Isolation, Durability). Atomicity memiliki pengertian transaksi merupakan sebuah unit satuan pekerjaan walaupun dalam keberjalanannya berhasil atau tidak berhasil. Consistency dalam hal ini memberikan pengertian referensi pada basis data, seperti *primary key* pada tabel dan lainnya. Isolation dibutuhkan apabila secara bersamaan terjadi proses transaksi, setiap transaksi harus terisolasi masing-masing untuk mencegah rusaknya data. Durability ialah setiap sebuah transaksi berhasil diselesaikan, hasil dari transaksi tersebut menjadi permanen dan tidak dapat dihapus dari basis data jika sistem mengalami kerusakan.

Penerapan 4 komponen dalam basis data dengan menggunakan SQL, secara mudah dapat dijelaskan menjadi 3 poin, yaitu sebagai berikut.

- Transaksi dimulai dengan menggunakan perintah *begin transaction.*
- Menjalankan aksi hapus, perbaharui atau tambah data menggunakan *query* SQL.
- Jika seluruh operasi berhasil, maka akan menjalankan perintah *commit*. Apabila operasi tidak berhasil, maka transaksi akan melakukan *rollback* pada seluruh operasi.

Pada kerangka kerja Spring terdapat *abstract layer* diatas API manajemen transaksi yang berbeda-beda. Tujuan adanya

transaksi pada Spring adalah untuk menyediakan alternatif pada transaksi EJB (Enterprise Java Beans) dengan menambahkan kemampuan transaksi pada POJO (Plain Old Java Object). EJB membutuhkan *server* aplikasi, sedangkan manajemen transaksi pada Spring dapat diimplementasikan tanpa membutuhkan *server* aplikasi.

2.6.3 Servlet

Servlet merupakan kelas dalam bahasa pemrograman Java yang digunakan untuk memperluas kemampuan *server* yang diakses oleh *host application*, dengan cara model pemrograman *request-response*. Walaupun *servlet* dapat merespon berbagai tipe *request*, biasanya *servlet* digunakan untuk memperluas aplikasi *host* di *web server*. Pada beberapa aplikasi, teknologi Java Servlet mendefinisikan kelas *servlet HTTP-Specific*.

Paket "javax.servlet" dan "javax.servlet.http" menyediakan *interface* dan kelas-kelas untuk mengimplementasikan *servlet*. Seluruh *servlet* harus mengimplementasikan *interface servlet* yang sekaligus mendefinisikan metode *life-cycle*. Saat melakukan implementasi *service* yang sifatnya umum, dapat digunakan atau memperluas kelas "GenericServlet" yang menyediakan API Java Servlet. Pada kelas "HttpServlet" terdapat *method* seperti "doGet" dan "doPost" untuk mengelola *HTTP-specific service*.

2.6.4 Beans

Kumpulan objek yang menjadi pembentuk aplikasi dan dikelola oleh wadah Spring IoC (Inverse of Control) sekarang bernama Dependency Injection adalah *beans*. Sebuah *bean* merupakan objek yang dipakai, dibentuk dan lainnya, dikelola oleh wadah Spring IoC. *Beans* ini dibuat dengan konfigurasi metadata yang diberikan pada wadah. Sebagai contoh, pada *file* XML terdapat definisi "<bean/>" yang berisikan informasi disebut metadata konfigurasi. Metadata konfigurasi dibutuhkan untuk

mengetahui bagaimana pembuatan sebuah *bean*, rincian *lifecycle bean* dan *dependency* dari *bean*.

Tiga komponen *method* penting untuk menyediakan metadata konfigurasi pada Spring Container, terdiri dari *file* konfigurasi berbasis XML, konfigurasi berbasis anotasi dan konfigurasi berbasis Java.

2.7 PostgreSQL

PostgreSQL atau sering disebut Postgres merupakan salah satu dari sejumlah basis data besar yang menawarkan skalabilitas, keluwesan, dan kinerja yang tinggi. Penggunaannya begitu meluas di berbagai *platform* dan didukung oleh banyak bahasa pemrograman. Bagi masyarakat TI (teknologi informasi) di Indonesia, Postgres sudah digunakan untuk berbagai aplikasi seperti *web*, sistem kasir, dan sistem informasi besar lainnya. Ada banyak hal unik yang bisa kita temui dari basis data yang satu ini. Niatan awal para *programmer*-nya adalah membuat suatu basis data yang kaya akan fitur dengan keluwesan yang tinggi.

Postgres menawarkan standar yang lebih baik. Dibalik masalah teknis tersebut, Postgres tersedia dalam bentuk kode sumber dan dapat diunduh tanpa pembebanan biaya. Tidak heran kalau Linux Award sempat menobatkan Postgres sebagai basis data pilihan yang diikuti Oracle sebagai *runner-up*-nya.

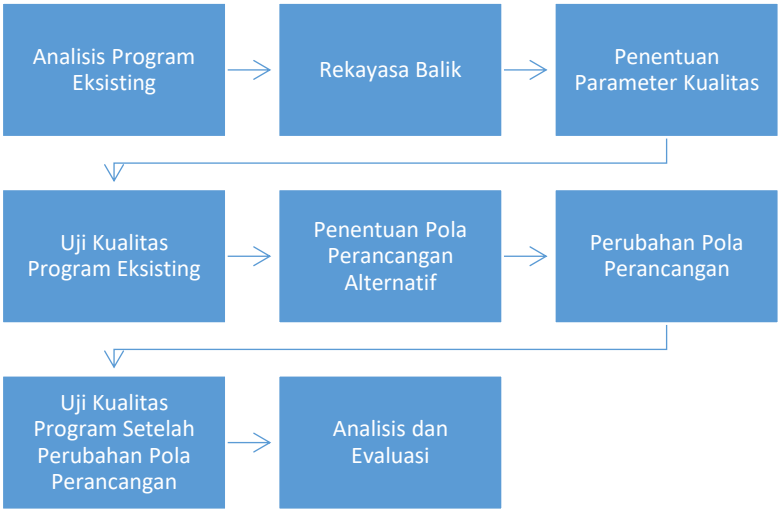
Perbedaan penting antara Postgres dengan sistem relasional standar adalah arsitektur Postgres yang memungkinkan user untuk mendefinisikan sendiri SQL-nya, terutama pada pembuatan fungsi atau biasa disebut sebagai *stored procedure*. Hal ini dimungkinkan karena informasi yang disimpan oleh Postgres bukan hanya tabel dan kolom, melainkan tipe, fungsi, metode akses, dan banyak lagi yang terkait dengan tabel dan kolom tersebut. Semuanya terhimpun dalam bentuk kelas yang bisa diubah pengguna. Arsitektur yang menggunakan kelas ini lazim disebut berorientasi objek. Karena Postgres bekerja dengan kelas, berarti Postgres lebih mudah dikembangkan di tingkat pengguna,

dan anda bisa mendefinisikan sebuah tabel sebagai turunan dari tabel lain. Sebagai perbandingan bahwa sistem basis data konvensional hanya dapat diperluas dengan mengubah kode sumbernya atau menggunakan modul tambahan yang ditulis khusus oleh *vendor*, maka dengan Postgres memungkinkan pengguna untuk membuat sendiri berkas objek atau *shared library* yang dapat diterapkan untuk mendefinisikan tipe data, fungsi, bahkan bahasa yang baru. Dengan demikian Postgres memiliki dua kekuatan besar yaitu kode sumber dan arsitektur yang luwes, tentunya disamping fitur penting lainnya seperti dokumentasi yang lengkap, dan sebagainya. Selain itu Postgres juga didukung oleh banyak antarmuka-antarmuka ke berbagai bahasa pemrograman seperti C++, Java, Perl, PHP, Python dan Tcl. ODBC dan JDBC juga tersedia yang membuat Postgres lebih terbuka dan dapat diterapkan secara meluas.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai alur pengerjaan tugas akhir. Alur pengerjaan tugas akhir ditunjukkan pada gambar Gambar 3.1. Ada 8 langkah yang harus dilakukan selama proses pengerjaan tugas akhir. Setiap langkah mempunyai *output* masing-masing yang diperlukan untuk langkah-langkah selanjutnya.



Gambar 3.1 Alur pengerjaan tugas akhir

Gambar 3.1 menunjukkan bahwa pengerjaan tugas akhir ini memiliki 8 langkah. Berikut adalah penjelasan dari langkah-langkah pengerjaan tugas akhir:

1. Analisis Program Eksisting

   Analisis program eksisting dilakukan dengan cara mempelajari modul Ekuivalensi awal yang telah

diselesaikan pengembangnya. Tujuan dari analisis program eksisting adalah untuk mendapatkan gambaran seperti apa struktur, fitur, serta pola perancangan yang telah diterapkan pada modul Ekuivalensi. Detail dari langkah ini akan dijelaskan pada sub-bab 3.1.

2. Rekayasa Balik

   Langkah berikutnya adalah melakukan rekayasa balik untuk mengetahui struktur modul Ekuivalensi. Tujuan utama dari langkah ini adalah untuk mendapatkan diagram kelas untuk mengetahui struktur dari kelas-kelas pada modul Ekuivalensi. Detail dari langkah ini akan dijelaskan di sub-bab 3.2.

3. Penentuan Parameter Kualitas

   Langkah selanjutnya yaitu menentukan parameter kualitas dari ISO 25023 yang akan digunakan untuk pengukuran kualitas baik sebelum ataupun sesudah perubahan pola perancangan. Pada langkah ini akan didapatkan parameter kualitas yang digunakan untuk mengukur kualitas modul Ekuivalensi. Penentuan parameter akan dijelaskan pada sub-bab 3.3.

4. Uji Kualitas Program Eksisting

   Setelah langkah sebelumnya yaitu menentukan parameter kualitas, maka uji kualitas program eksisting akan dilakukan dengan parameter-parameter tersebut. Pada langkah ini akan didapatkan hasil kualitas berupa angka-angka yang didapatkan sesuai rumus yang telah disediakan oleh ISO 25023. Rumus dan hasil uji kualitas akan dijelaskan pada Bab 5.

5. Penentuan Pola Perancangan Alternatif

   Langkah selanjutnya yaitu menentukan pola perancangan yang dipilih untuk mengubah pola perancangan program eksisting. Pada langkah ini akan didapatkan pola perancangan yang akan digunakan untuk mengubah struktur modul Ekuivalensi. Pola perancangan alternatif terpilih akan dijelaskan pada sub-bab 3.4.

6. Perubahan Pola Perancangan

   Setelah pola perancangan ditentukan pada langkah sebelumnya, program eksisting akan langsung diubah dengan mengimplementasikan pola-pola perancangan yang dipilih. Pada langkah ini akan didapatkan modul Ekuivalensi yang telah diubah sesuai pola perancangan yang dipilih. Detail perubahan akan dijelaskan pada Bab 4.
7. Uji Kualitas Program Setelah Perubahan Pola Perancangan

   Setelah menyelesaikan perubahan pola perancangan, kualitas modul Ekuivalensi kembali dihitung dengan parameter kualitas yang sudah dipilih saat menentukan parameter kualitas. Pada langkah ini akan didapatkan angka-angka hasil pengukuran kualitas modul Ekuivalensi setelah perubahan pola perancangan. Hasil uji kualitas setelah perubahan pola perancangan akan dijelaskan pada Bab 5.
8. Analisis dan Evaluasi

   Pada analisis dan evaluasi, hasil penghitungan kualitas yang baru akan dibandingkan dengan hasil penghitungan kualitas program eksisting. Setelah itu akan dilakukan analisis beserta evaluasi yang akan dijelaskan di Bab 5.

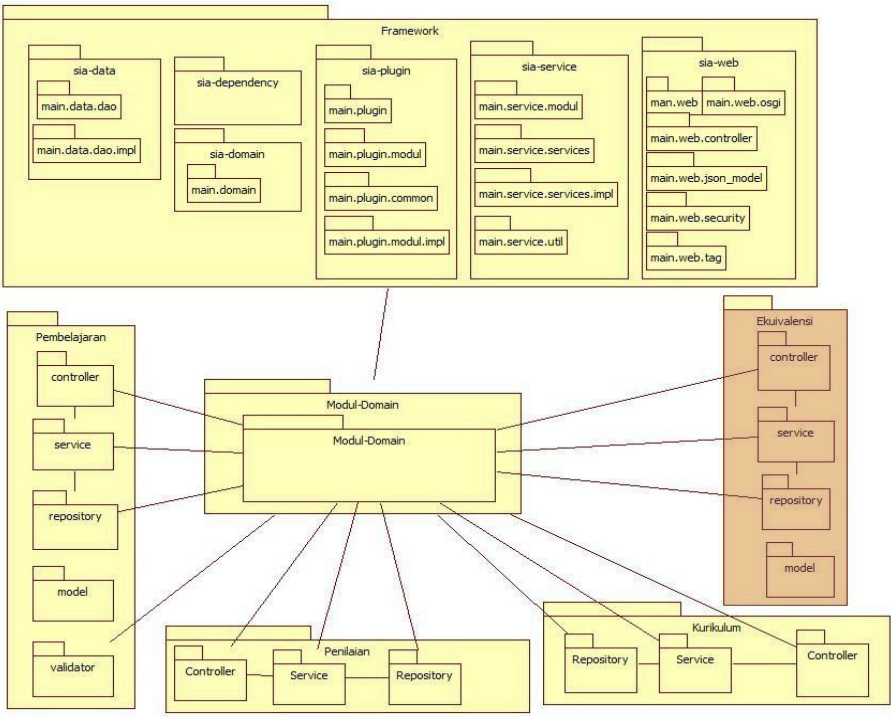
3.1 Analisis Program Eksisting

Sistem informasi akademik yang digunakan adalah sistem informasi akademik versi penelitian. Sistem informasi akademik ini terdiri dari 6 modul yaitu modul Framework, Domain, Pembelajaran, Ekuivalensi, Kurikulum, dan Penilaian. Modul Framework adalah modul pusat, Modul Domain sebagai penghubung modul lain dengan basis data, dan 4 modul lain adalah modul yang menjalankan fungsi akademik.

Sistem informasi akademik ini menggunakan bahasa pemrograman Java dengan Spring MVC untuk *web development*. Selain itu arsitektur modul didukung dengan menggunakan Eclipse

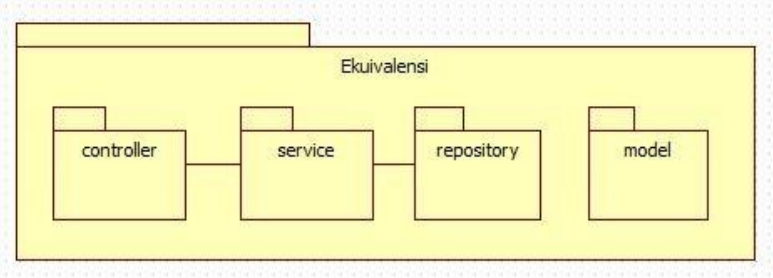
Virgo dan OSGI Framework. Web Server yang digunakan adalah Tomcat. Berikutnya akan dijelaskan mengenai arsitektur, fitur dan hak aksesnya, serta pola perancangan yang sudah diimplementasikan dalam sistem informasi akademik.

3.1.1 Diagram Paket



Gambar 3.2 Diagram paket sistem informasi akademik

Gambar 3.2 menjelaskan bahwa sistem informasi akademik terbagi menjadi 6 modul yaitu modul Framework, Domain, Ekuivalensi, Kurikulum, Penilaian, dan Pembelajaran. Topik tugas akhir ini akan berfokus pada modul Ekuivalensi.



Gambar 3.3 Diagram Paket modul Ekuivalensi

Pada Gambar 3.3 dijelaskan mengenai arsitektur utama dari modul Ekuivalensi yaitu:

a. Controller

   Paket ini berisi *class* Controller untuk melakukan *request mapping* sehingga bekerja sesuai URL yang diterima.

b. Model

   Pada modul Ekuivalensi, model hanya berisi satu *file* yaitu FileUpload.java yang berguna untuk melakukan *upload file*.

c. Repository

   Paket ini berisi *class* untuk melakukan fungsi *query* dasar ke basis data seperti *create, get, update*, dan *delete*. Mayoritas *file* pada paket Repository adalah *class* Interface dan implementasinya.

d. Service

   Paket ini berisi *class* yang menghubungkan Controller dan Repository. Paket ini berisi *class* yang memperantarai Controller dan Repository dalam beberapa *logic* dan akses ke basis data.

3.1.2 Hak Akses dan Fitur

Berikut hak akses sekaligus fitur fitur yang ada pada modul Ekuivalensi:

Tabel 3.1 Hak Akses dan Fitur

Hak Akses	Fitur
Tim Ekuivalensi	Verifikasi Ekuivalensi
	Kelola Aturan Ekuivalensi
	Verifikasi Mata Kuliah Wajib
	Kelola Ekuivalensi Peserta Didik Alih Jenjang
Admin	Kelola Katalog Alih Jenjang
	Kelola Mata Kuliah Alih Jenjang
	Bagikan Katalog ke Satuan Manajemen
	Kelola Peserta Didik Alih Jenjang
	Kelola Aturan Ekuivalensi Alih Jenjang
	Cetak Laporan Ekuivalensi Alih Jenjang
Peserta Didik	Cetak Laporan Ekuivalensi

3.1.3 Pola Perancangan dari Program Eksisting

Pola perancangan yang saat ini digunakan oleh modul Ekuivalensi di sistem informasi akademik adalah:

1. Model View Controller

   Model View Controller (MVC) sudah digunakan dengan sendirinya karena penggunaan Framework Spring MVC. Perbedaannya adalah model yang dibagi-bagi menjadi beberapa bagian yaitu Service, Repository, dan Domain. Hal ini dipengaruhi oleh arsitektur dan adanya implementasi pola perancangan lain.

2. Service Layer

   Pada program eksisting, Service Layer diimplementasikan dengan *class* Service yang menjadi

perantara dari Controller dan Repository. Di dalam Service terdapat beberapa *business logic*.

3. Domain Model

   Modul Ekuivalensi menggunakan Domain dari modul Sia-modul-domain. Domain Model digunakan oleh Controller, Service dan Repository sebagai model objek dan digunakan sebagai representasi tabel di basis data beserta kolomnya yang akan menerima hasil *query* dari Repository. Domain yang telah terisi data hasil *query* tersebut akan dikirim ke Service dan Controller.

3.2 Rekayasa Balik

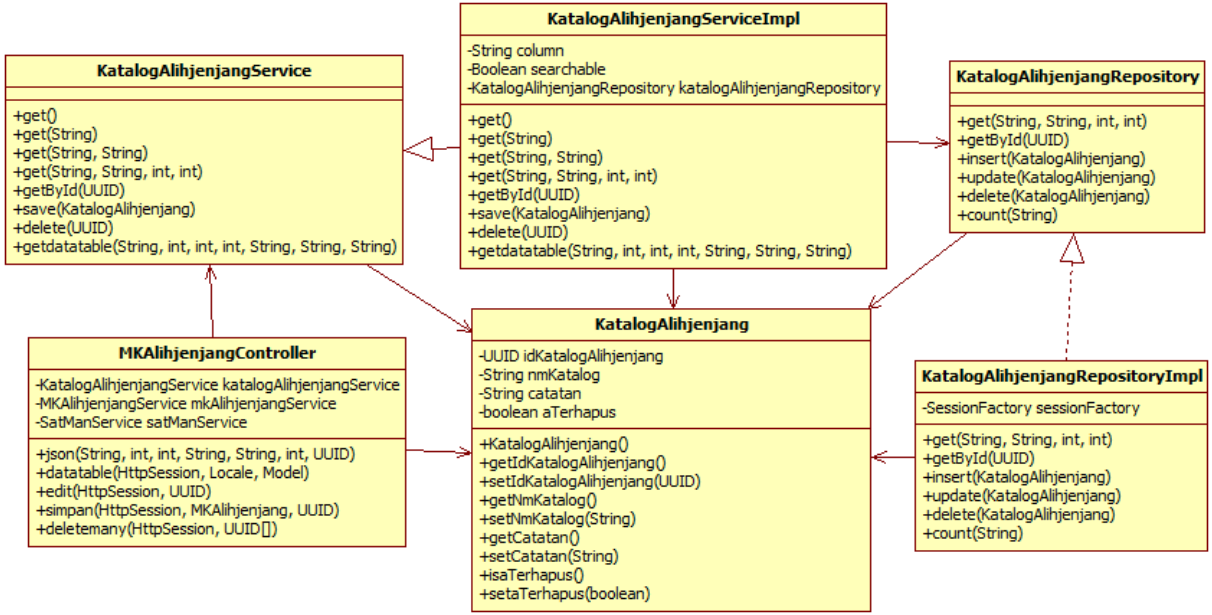
Rekayasa balik dilakukan dengan membuat diagram kelas. Diagram kelas dari salah satu bagian program eksisting dapat dilihat pada Gambar 3.4. Gambar 3.4 menunjukkan bahwa pola perancangan utama yang digunakan oleh program eksisting adalah Service Layer. MKAlihjenjangController berfungsi sebagai Controller untuk memproses *request*. KatalogAlihjenjangService adalah sebuah Service yang berfungsi sebagai perantara antara Controller dan Repository sehingga Controller dapat melakukan fungsi yang mengakses basis data. KatalogAlihjenjangRepository adalah sebuah Repository berisi fungsi-fungsi yang terhubung ke basis data. Pada Service dan Repository, setiap *class* utama terdiri dari 2 bagian *class*, yaitu Interface dan implementasinya. *Class* KatalogAlihjenjang adalah Domain berfungsi sebagai representasi tabel pada basis data yang berisi fungsi *setter* dan *getter*.

Tabel 3.2 menunjukkan jumlah *class* yang digunakan oleh modul Ekuivalensi. Pada Tabel 3.2 terlihat adanya paket com.bustan.siakad.model. Paket ini hanya berisi sebuah *class* yaitu FileUpload.java. Namun setelah ditelusuri, FileUpload.java sama sekali tidak digunakan oleh siapapun sehingga *class* ini adalah sebuah *dead class* dan tidak akan muncul dalam hasil rekayasa balik. FileUpload.java juga akan dihilangkan pada implementasi pola perancangan alternatif.

Tabel 3.2 Jumlah *class* pada pola perancangan awal

Paket	**Jumlah *Class***
com.bustan.siakad.controller	7
com.bustan.siakad.model	1
com.bustan.siakad.service	57
com.bustan.siakad.repository	54
com.sia.modul.domain	34

Contoh Implementasi pola perancangan awal :



Gambar 3.4 Contoh diagram kelas pola perancangan awal

3.3 Penentuan Parameter Kualitas

Parameter kualitas yang dipilih adalah berdasarkan ISO 25023. Parameter yang dipilih yaitu 2 bagian dalam Usability dan 4 bagian dalam Maintainability. Usability dan Maintainability dipilih karena pengukuran kualitas berfokus pada kualitas internal terutama pada internal *source code*.

3.3.1 Usability

Tabel 3.3 Usability

Sub-kualitas	Kode Poin Penilaian	Nama Poin Penilaian
Appropriateness Recognisability	UAp-1-G	Description Completeness
	UAp-2-S	Demonstration Capability
	UAp-3-S	Entry Point Self-descriptiveness
Operability	UOp-1-G	Operational Consistency
	UOp-2-G	Message Clarity
	UOp-3-S	Functional Customizability
	UOp-4-S	User Interface Customizability
	UOp-5-S	Monitoring Capability
	UOp-6-S	Undo Capability
	UOp-7-S	Terminology Understandability
	UOp-8-S	Appearance Consistency

Tabel 3.3 menunjukkan bahwa ada 2 sub-kualitas yang digunakan pada parameter kualitas Usability yaitu Appropriateness Recognisability dan Operability. Sub-kualitas Appropriateness Recognisability pada intinya digunakan untuk mengukur sejauh mana pengguna dapat mengetahui apakah sebuah produk atau

sistem sesuai dengan kebutuhannya. Appropriateness Recognisability memiliki 3 poin kualitas. Sub-kualitas Operability pada intinya mengukur sejauh mana sebuah sistem memiliki atribut-atribut yang membuat sistem tersebut mudah digunakan. Operability memiliki 8 poin kualitas.

Appropriateness Recognisability dan Operability dipilih karena diperkirakan kualitasnya akan dipengaruhi oleh perubahan *source code*. Perubahan *source code* terjadi karena diimplementasikannya perubahan pola perancangan. Sedangkan sub-kualitas lain pada Usability yaitu Learnability, User Error Protection, User Interface Aesthetics, dan Accessibility tidak dipilih karena diketahui sejak awal bahwa 4 sub-kualitas ini tidak akan dipengaruhi oleh pola perancangan yang memodifikasi *source code*.

3.3.2 Maintainability

Tabel 3.4 Maintainability

Sub Kualitas	Kode Poin Penilaian	Nama Poin Penilaian
Modularity	MMo-1-G	Coupling of Components Conformance
	MMo-2-S	Cyclomatic Complexity
Reusability	MRe-1-G	Reusability of Assets
	MRe-2-S	Conformance to Coding Rules
Analysability	MAn-1-G	System Log Completeness Conformance
	MAn-2-S	Diagnosis Function Effectiveness
	Man-3-S	Diagnosis Function Sufficiency
Testability	MTe-1-G	Test Function Completeness Conformance
	MTe-2-S	Autonomous Testability
	MTe-3-S	Test Restartability

Tabel 3.4 menunjukkan bahwa ada 4 sub-kualitas yang digunakan pada parameter kualitas Maintainability yaitu Modularity, Reusability, Analysability, dan Testability. Modularity digunakan untuk mengukur sejauh mana sebuah perubahan sebuah komponen pada suatu sistem memiliki pengaruh minimal ke komponen lain. Reusability digunakan untuk mengukur apakah suatu aset di dalam suatu sistem dapat digunakan oleh sistem atau aset lain. Analysability mengukur apakah memungkinkan untuk menilai suatu dampak perubahan pada produk atau sistem, atau untuk mendiagnosis kekurangan dan kegagalan produk yang dapat terjadi, atau untuk mengidentifikasi bagian yang akan dimodifikasi. Testability mengukur sejauh mana kriteria sebuah tes dapat diterapkan ke sebuah sistem dan apakah beberapa tes bisa dilakukan untuk memenuhi kriteria-kriteria tersebut.

Sama seperti parameter kualitas Usability, pemilihan 4 sub-kualitas pada parameter kualitas Maintainability yaitu Modularity, Reusability, Analysability, dan Testability disebabkan karena 4 sub-kualitas tersebut akan dipengaruhi oleh pola perancangan yang memodifikasi *source code*. Selain itu karena waktu pengerjaan tugas akhir yang terbatas, 4 sub-kualitas tersebut juga tidak membutuhkan waktu dan usaha yang banyak untuk proses pengukurannya. Sementara satu sub-kualitas lagi pada Maintainability yaitu Modifiability tidak dipilih karena meskipun ada kemungkinan dapat dipengaruhi pola perancangan yang mengubah *source code*, namun membutuhkan waktu dan proses yang tidak mencukupi untuk melakukan pengukuran. Analisis dari poin kualitas dari seluruh sub-kualitas yang dipilih akan dijelaskan di Bab 5.

3.4 Penentuan Pola Perancangan Alternatif

Pola perancangan alternatif yang akan diterapkan ke sistem informasi akademik adalah Active Record dan Data Mapper. Pada program eksisting, *class* Repository sebagai *data source* yang

berinteraksi dengan basis data tidak mengimplementasikan pola-pola perancangan yang termasuk dalam Data Source Pattern [12]. Karena itu, Active Record dan Data Mapper dipilih sebagai pola perancangan alternatif karena termasuk dalam Data Source Pattern dan dapat diimplementasikan ke program eksisting.

3.4.1 Active Record

Active Record [12] adalah kelas yang isinya adalah data dan tingkah laku dari data-data tersebut. Dalam implementasinya, sebuah *class* Active Record akan berisi data-data beserta perintah-perintah logic yang menggunakan data-data tersebut. Active Record dipilih untuk perubahan pola perancangan karena program eksisting dari modul Ekuivalensi diketahui hanya memiliki perintah-perintah *logic* yang tidak terlalu kompleks seperti *read*, *get*, *insert*, *update*, dan *delete* sehingga sangat cocok dengan sifat Active Record yang direkomendasikan untuk sistem yang sederhana.

3.4.2 Data Mapper

Data Mapper [12] memisahkan antara *domain* dan *data source* sehingga perpindahan data antara objek dan basis data menjadi independen. Pemisahan dilakukan dengan membuat objek baru bernama Mapper yang bertugas memetakan *logic*. Data Mapper dipilih karena penggunaan *class* Service dan Repository yang mirip dengan Mapper yang memetakan *logic* sesuai data yang diperlukan.

[Halaman ini sengaja dikosongkan]

.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi perubahan pola perancangan terhadap sistem. Sistem akan diubah dengan 2 pola yaitu Active Record dan Data Mapper. Bahasa pemrograman yang digunakan untuk implementasi adalah bahasa pemrograman Java menggunakan Framework Spring.

4.1 Implementasi Active Record

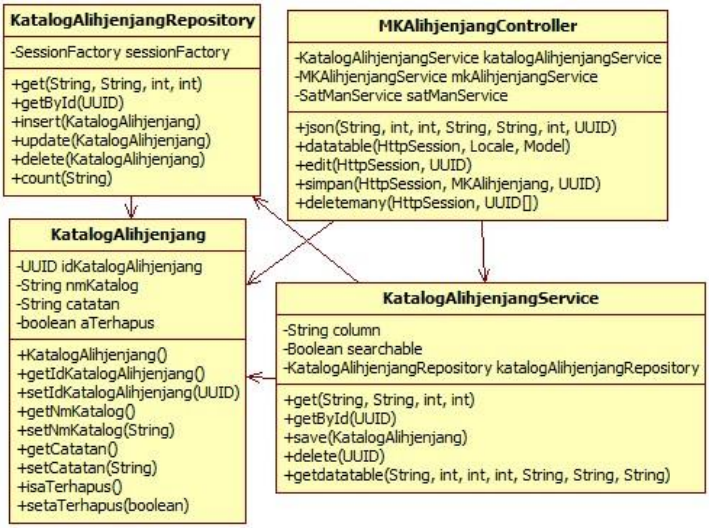
Pada pola perancangan program eksisting, representasi table, akses ke basis data, dan *logic* dibuat terpisah dalam 3 jenis *class* yaitu Service, Repository, dan Domain. Service berisi beberapa *logic* dimana akan memerlukan Repository saat membutuhkan data dari basis data. Repository berisi akses ke basis data dan beberapa *query* dasar ke basis data. Domain bertindak sebagai representasi tabel dan kolomnya beserta *setter* dan *getter* untuk setiap kolom. Untuk contoh pola perancangan program eksisting yang akan digunakan untuk perbandingan dapat dilihat pada Gambar 3.4.

Inti dari pengubahan pola perancangan program eksisting menjadi Active Record adalah penyatuan peran utama dari Service, Repository, dan Domain. Langkah-langkah yang dilakukan untuk mengubah pola perancangan program eksisting menjadi Active Record adalah sebagai berikut:

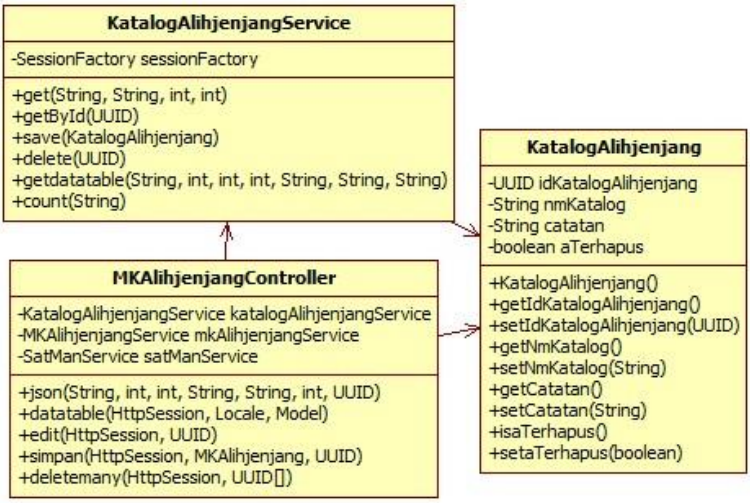
1. Menghapus *class* Interface dan langsung menggunakan implementasinya. Penghapusan Interface dilakukan karena penggunaan *class* Interface yang ada pada modul Ekuivalensi sangat tidak efektif. Hasilnya dapat dilihat di Gambar 4.1.
2. Memindahkan isi seluruh *method* dari semua Repository ke dalam Service yang sesuai untuk menggantikan pemanggilan Repository oleh Service.

3. Menghapus semua paket Repository beserta seluruh *class* didalamnya. Hasilnya dapat dilihat di Gambar 4.2.
4. Memindahkan seluruh *method* dari semua Service ke Domain yang sesuai sehingga Domain berubah menjadi sebuah *class* Active Record karena merupakan representasi tabel dan kolomnya, mempunyai *setter* dan *getter* dari setiap kolom, mempunyai akses ke basis data, berisi *query* ke basis data, dan berisi *logic* yang bisa digunakan oleh Controller.
5. Menghapus semua paket Service beserta seluruh *class* didalamnya.
6. Mengubah pemanggilan Service pada Controller menjadi pemanggilan *class* Active Record yang sesuai.

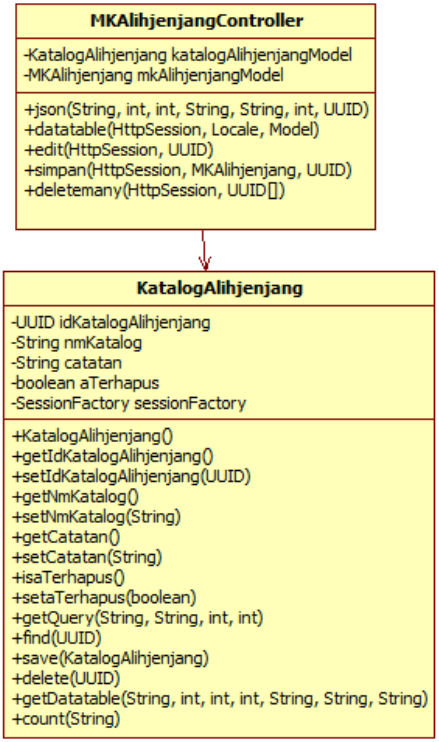
Hasil perubahan pola perancangan ke Active Record dapat dilihat pada Gambar 4.3 dan dapat dibandingkan dengan pola perancangan awal pada Gambar 3.4. Sesuai dengan Gambar 4.3, pada *class* Katalogalihjenjang terdapat representasi kolom dari basis data, *setter* dan *getter* untuk setiap kolom, *query* ke basis data, dan *logic* yang dapat digunakan oleh Controller. Dibandingkan pola perancangan awal di Gambar 3.4, Active Record lebih sederhana dikarenakan penyatuan Service, Repository, dan Domain. Active Record sangat cocok untuk modul Ekuivalensi karena sama sekali tidak ada relasi tabel ataupun *query* ke basis data yang bersifat sangat kompleks.



Gambar 4.1 Hasil penghapusan Interface



Gambar 4.2 Hasil penghapusan Repository



Gambar 4.3 Diagram kelas dari Active Record

Tabel 4.1 Perbandingan *class* sebelum dan sesudah implementasi Active Record

***Class* pada Program Eksisting**	**Hasil Perubahan Active Record**
KatalogAlihjenjangService	Dihapus karena merupakan sebuah Interface
KatalogAlihjenjangServiceImpl	Menerima seluruh *method* dari RepositoryImpl, setelah itu seluruh *method* dipindahkan ke *class* KatalogAlihjenjang yang merupakan *class* Active

***Class* pada Program Eksisting**	**Hasil Perubahan Active Record**
	Record. Setelah itu *class* ini dihapus
KatalogAlihjenjangRepository	Dihapus karena merupakan sebuah Interface
KatalogAlihjenjangRepositoryImpl	Semua *method* dipindahkan ke ServiceImpl, lalu *class* ini dihapus
MKAlihjenjangController	Penghapusan *import* Service dan Repository
KatalogAlihjenjang	Menerima semua *method* dari ServiceImpl dan RepositoryImpl dan menjadi sebuah *class* Active Record

Tabel 4.1 menunjukkan contoh perubahan yang terjadi pada setiap *class* dari program eksisting menjadi Active Record. Seluruh *class* Service dan Reposity mengalami penghapusan Interface dan implementasinya mengalami perpindahan *method* lalu setelah itu dihapus. Tabel 4.2 menunjukkan bahwa total jumlah *class* menjadi lebih sedikit dibanding pola perancangan awal sehingga dapat menghemat *space* meskipun dalam jumlah yang kecil. Jumlah *class* yang lebih sedikit selain karena penyatuan juga disebabkan oleh dihapusnya *class* yang sama sekali tidak terpakai.

Tabel 4.2 Jumlah *class* pada implementasi Active Record

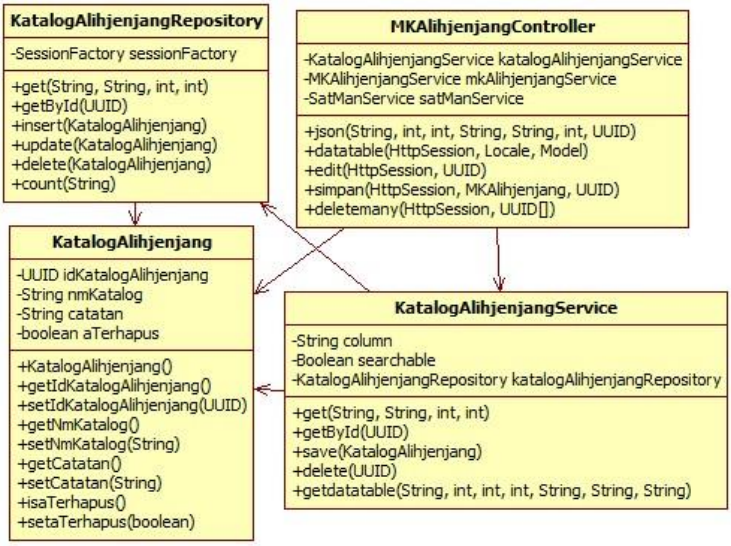
Paket	**Jumlah *Class***
com.bustan.siakad.controller	7
com.sia.modul.domain	37

4.2 Implementasi Data Mapper

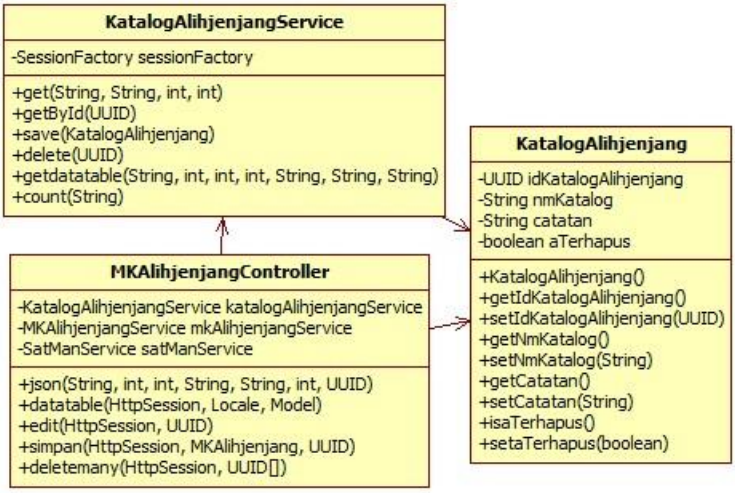
Pada perubahan pola perancangan program eksisting ke pola perancangan Data Mapper, inti dari perubahan yang dilakukan adalah penggabungan fungsi-fungsi utama pada Service dan Repository pada pola perancangan program eksisting menjadi sebuah *class* baru yaitu *class* Mapper. Domain yang digunakan sama dengan digunakan pada pola perancangan awal dan tidak akan diubah seperti yang terjadi di Active Record sebelumnya. Untuk contoh pola perancangan program eksisting yang akan digunakan untuk perbandingan dapat dilihat pada Gambar 3.4.

Langkah-langkah yang dilakukan untuk mengubah pola perancangan program eksisting menjadi Data Mapper adalah sebagai berikut:

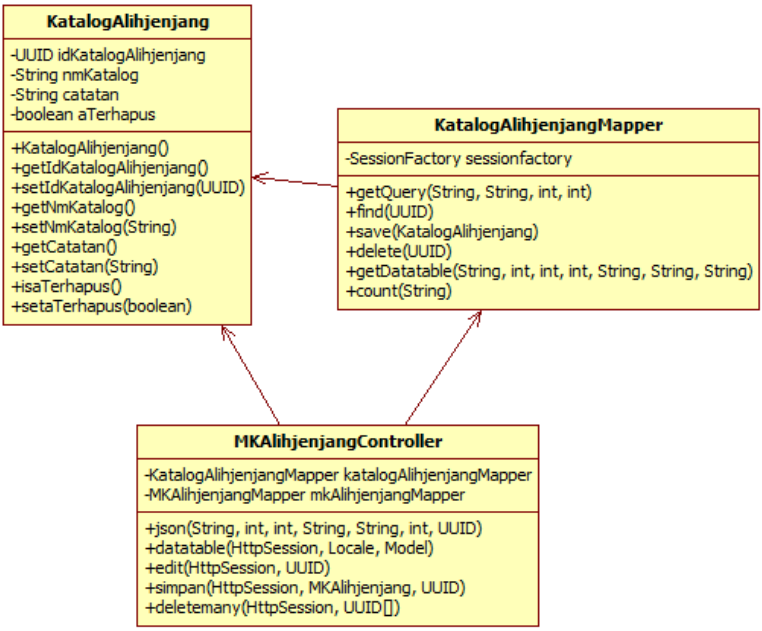
1. Sama seperti saat perubahan ke Active Record, yang dilakukan pertama adalah menghapus *class* Interface dan langsung menggunakan implementasinya. Penghapusan Interface dilakukan karena penggunaan *class* Interface yang ada pada modul Ekuivalensi sangat tidak efektif. Hasilnya dapat dilihat di Gambar 4.4.
2. Memindahkan isi seluruh *method* dari semua Repository ke dalam Service yang sesuai untuk menggantikan pemanggilan Repository oleh Service.
3. Menghapus semua paket Repository beserta seluruh *class* didalamnya. Hasilnya dapat dilihat di Gambar 4.5.
4. Mengganti nama *class* seluruh Service menjadi Mapper.
5. Mengganti nama paket Service menjadi Mapper.
6. Mengubah pemanggilan Service pada Controller menjadi pemanggilan *class* Mapper yang sesuai.



Gambar 4.4 Hasil penghapusan Interface



Gambar 4.5 Hasil penghapusan Repository



Gambar 4.6 Diagram kelas Data Mapper

Hasil perubahan dapat dilihat di Gambar 4.6. Gambar 4.6 menunjukkan bahwa MKAlihjenjangController menggunakan KatalogAlihjenjangMapper sebagai perantara untuk mendapatkan hasil *query* dari basis data. MKAlihjenjang Controller juga menggunakan KatalogAlihjenjang sebagai penerima hasil *query* dari Mapper. KatalogAlihjenjangMaper menggunakan KatalogAlihjenjang sebagai tipe data yang menerima *query* dari basis data dan mengirimnya ke Controller. Gambar 4.6 juga menunjukkan kelemahan Data Mapper yaitu ketergantungan Mapper KatalogAlihjenjangMapper dan model objek KatalogAlihjenjang yang tidak sesuai dengan arsitektur *three-tier* karena *upward dependency* antara Mapper di *domain layer* dan model objek di *presentation layer*.

Tabel 4.3 Perbandingan *class* sebelum dan sesudah implementasi Data Mapper

***Class* pada Program Eksisting**	**Hasil Perubahan Data Mapper**
KatalogAlihjenjangService	Dihapus karena merupakan sebuah Interface
KatalogAlihjenjangServiceImpl	Menerima seluruh *method* dari RepositoryImpl, setelah itu *class* ini menjadi sebuah *class* Mapper dengan nama KatalogAlihjenjangMapper
KatalogAlihjenjangRepository	Dihapus karena merupakan sebuah Interface
KatalogAlihjenjangRepositoryImpl	Semua *method* dipindahkan ke ServiceImpl, lalu *class* ini dihapus
MKAlihjenjangController	Penghapusan *import* Service dan Repository. Setelah itu melakukan *import* ke Data Mapper yang diperlukan
KatalogAlihjenjang	Tidak berubah

Tabel 4.3 menunjukkan contoh perubahan yang terjadi pada setiap *class* dari program eksisting menjadi Data Mapper. Seluruh *class* Service dan Reposity mengalami penghapusan Interface Sementara *method* dari implementasi Repository dipindahkan ke implementasi Service dan setelah itu implementasi Service menjadi sebuah class Mapper.

Tabel 4.4 menunjukkan bahwa jumlah sama seperti Active Record, jumlah *class* juga berkurang. Hal ini dikarenakan *class* Service dan *class* Repository pada perancangan awal disatukan menjadi *class* Mapper. Selain itu juga adanya penghapusan *class* yang tidak terpakai. Lalu terdapat sebuah paket baru yaitu paket Libraries yang berisi representasi sebuah tipe data yang mirip dengan Domain namun tidak merepresentasikan tabel dan tidak mengakses basis data sehingga berada di luar perubahan pola

perancangan. Semua *class* yang ada pada paket Libraries sebelumnya berada dalam paket Service.

Tabel 4.4 Jumlah *class* pada implementasi Data Mapper

Paket	Jumlah *Class*
com.bustan.siakad.controller	7
com.bustan.siakad.mapper	23
com.sia.modul.domain	34
com.sia.bustan.libraries	3

4.3 Permasalahan Saat Implementasi

Saat pengimplementasian perubahan pola perancangan tentunya mustahil untuk tidak menemui masalah. Permasalahan yang ditemui sangat memperlambat waktu implementasi dari yang telah diperkirakan. Berikut masalah-masalah yang ditemui:

1. Minimnya dokumentasi mengenai konfigurasi untuk menjalankan program eksisting sistem informasi akademik terutama untuk Virgo dan Eclipse IDE. Hal ini membuat terpakainya waktu yang cukup lama hanya untuk dapat menjalankan program eksisting. Minimnya dokumentasi juga membuat program eksisting tidak bisa dijalankan di *server* lokal sehingga untuk mencoba setiap perubahan diharuskan untuk terkoneksi ke *server* utama. Masalah ini diatasi dengan beberapa *trial and error*, menggunakan *server* utama untuk melakukan tes, dan tidak membuat *server* lokal.
2. Banyak *dependency* Java untuk program eksisting yang akses untuk mendapatkannya sudah berubah atau kedaluwarsa sehingga mempersulit untuk melakukan *build* modul Ekuivalensi ke *file* JAR. Masalah ini diatasi dengan mencari *dependency* secara manual dengan mencari sumber eksternal di luar sumber asli *dependency*.

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan uji kualitas yang dilakukan pada modul Ekuivalensi. Pembahasan pengujian meliputi analisis pengukuran kualitas dan hasil pengukuran sebelum dan sesudah perubahan pola perancangan.

5.1 Analisis Pengukuran Kualitas

Pengukuran kualitas yang digunakan adalah berdasarkan ISO 25023. Setiap poin kualitas yang ada pada setiap sup-kualitas dari Usability dan Maintainability memiliki rumus untuk penghitungan kualitas mereka masing-masing. Setiap rumus menggunakan 2 variabel, yaitu A dan B. Nilai A dan B berbeda pada masing-masing poin kualitas. Nilai A dan B dicari sendiri sesuai petunjuk yang ada pada setiap poin kualitas. Setelah nilai A dan B diketahui, maka nilai A dan B dapat digunakan pada rumus yang telah disediakan pada setiap poin kualitas. Hasil dari rumus yang digunakan nilainya akan selalu berada diantara atau tepat di 0 dan 1.

5.1.1 Usability

Usability mengukur sejauh mana sebuah produk atau sistem dapat digunakan oleh pengguna yang dituju untuk mencapai suatu tujuan dengan efektif dan efisien. Dalam tugas akhir ini dipilih 2 sub-kualitas dalam Usability yaitu Aprropriateness Recognisability dan Operability.

5.1.1.1 Appropriateness Recognisability

Sub-kualitas Appropriateness Recognisability pada intinya digunakan untuk mengukur sejauh mana pengguna dapat mengetahui apakah sebuah produk atau sistem sesuai dengan kebutuhannya. Appropriateness Recognisability memiliki 3 poin

kualitas yaitu Description Completeness, Demonstration Capability, dan Entry Point Self-descriptiveness.

Tabel 5.1 Description Completeness

Nama	Description Completeness
ID	UAp-1-G
Deskripsi	Proporsi dari skenario yang dideskripsikan pada deskripsi produk atau dokumen pengguna
Rumus	X = A / B
A	Jumlah skenario yang dideskripsikan pada deskripsi produk atau dokumen pengguna
B	Jumlah seluruh skenario produk
Penghitungan	Kasus penggunaan dan deskripsi kasus penggunaan pada buku TA [6]

Tabel 5.2 Demonstration Capability

Nama	Demonstration Capability
ID	UAp-2-S
Deskripsi	Proporsi dari fungsi yang mampu mendemonstrasikan kepada pengguna untuk menilai kelayakan
Rumus	X = A / B
A	Jumlah fitur dengan kemampuan demonstrasi
B	Jumlah fitur yang bisa diuntungkan dengan kemampuan demonstrasi
Penghitungan	Mengecek kemampuan demonstrasi setiap proses selama menjalankan fitur pada modul Ekuivalensi

Tabel 5.3 Entry Point Self-descriptiveness

Nama	Entry Point Self-descriptiveness
ID	UAp-3-S
Deskripsi	Proporsi dari *landing pages* dari *website* yang menjelaskan tujuan dari *website*
Rumus	X = A / B
A	Jumlah *landing pages* yang menjelaskan tujuan dari *website*
B	Jumlah seluruh *landing pages* pada *website*

Penghitungan	Mengecek halaman utama dari setiap proses utama

Tabel 5.1 menunjukkan bahwa Description Completeness dihitung dengan cara menghitung skenario produk yang ada pada dokumentasi. Dokumentasi yang digunakan adalah Buku Tugas Akhir [6]. Tabel 5.2 menunjukkan bahwa Demonstration Capability dihitung dengan melihat fitur yang dapat mendemonstrasikan dirinya sendiri ke pengguna. Tabel 5.3 menunjukkan bahwa Entry Point Self-descriptiveness diukur dengan cara melihat *landing pages* di modul Ekuivalensi.

5.1.1.2 Operability

Operability menilai seberapa mudah suatu produk atau sistem memiliki atribut yang membuat sistem atau produk tersebut mudah digunakan. Sub-kualitas Operability memiliki 8 poin kualitas yaitu Operational Consistency, Message Clarity, Functional Customizability, User Interface Customizability, Monitoring Capability, Undo Capability, Terminology Understandability, dan Appearance Consistency.

Tabel 5.4 Operational Consistency

Nama	Operational Consistency
ID	UOp-1-G
Deskripsi	Fungsi yang memiliki tampilan dan aturan yang konsisten dalam tugasnya
Rumus	X = 1 - A / B
A	Jumlah fitur yang inkonsisten
B	Jumlah fitur yang perlu konsistensi
Penghitungan	Mengecek konsistensi setiap proses utama pada fitur

Tabel 5.5 Message Clarity

Nama	Message Clarity
ID	UOp-2-G
Deskripsi	Proporsi pesan dari sistem yang dapat dimengerti
Rumus	X = A / B

A	Jumlah pesan yang memberikan instruksi atau hasil yang benar kepada pengguna
B	Jumlah seluruh pesan dalam sistem
Penghitungan	Mengecek pesan *pop-up* yang ada pada setiap proses

Tabel 5.6 Functional Customizability

Nama	Functional Customizability
ID	UOp-3-S
Deskripsi	Proporsi fitur dan prosedur yang bisa bebas diatur oleh pengguna
Rumus	X = A / B
A	Jumlah fitur dan prosedur yang bisa diatur oleh pengguna
B	Jumlah seluruh fitur dan operasi
Penghitungan	Mengecek setiap proses utama

Tabel 5.7 User Interface Customizability

Nama	User Interface Customizability
ID	UOp-4-S
Deskripsi	Proporsi elemen *user interface* yang tampilannya bisa diatur
Rumus	X = A / B
A	Jumlah elemen *user interface* yang tampilannya bisa diatur
B	Jumlah elemen *user interface* yang bisa diuntungkan dengan fungsi pengaturan
Penghitungan	Mengecek setiap halaman dari proses utama

Tabel 5.8 Monitoring Capability

Nama	Monitoring Capability
ID	UOp-5-S
Deskripsi	Proporsi *state* dari fitur yang bisa dimonitor saat dijalankan
Rumus	X = A / B
A	Jumlah fitur yang mempunyai *state* yang bisa dimonitor

B	Jumlah fitur yang bisa diuntungkan dengan monitoring
Penghitungan	Mengecek log dari setiap proses utama

Tabel 5.9 Undo Capability

Nama	Undo Capability
ID	UOp-6-S
Deskripsi	Proporsi dari fitur yang mempunyai opsi konfirmasi ulang atau *undo*
Rumus	X = A / B
A	Jumlah fitur yang mempunyai opsi konfirmasi ulang atau *undo*
B	Jumlah fitur yang bisa diuntungkan dengan opsi konfirmasi ulang atau *undo*
Penghitungan	Mengecek kemampuan konfirmasi ulang atau *undo* pada setiap proses utama

Tabel 5.10 Terminology Understandability

Nama	Terminology Understandability
ID	UOp-7-S
Deskripsi	Proporsi istilah-istilah dalam *user interface* yang dipahami oleh pengguna
Rumus	X = A / B
A	Jumlah istilah pada *user interface* yang bisa dimengerti oleh pengguna yang dituju
B	Jumlah istilah yang ada pada *user interface*
Penghitungan	Melihat kuesioner untuk klien pada dokumentasi Buku Tugas Akhir [6]

Tabel 5.11 Appearance Consistency

Nama	Appearance Consistency
ID	UOp-8-S
Deskripsi	Proporsi dimana *user interface* dengan isi sama mempunyai tampilan yang sama
Rumus	X = 1 - A / B
A	Jumlah *user interface* dengan isi sama namun mempunyai tampilan berbeda
B	Jumlah *user interface* dengan isi sama

Penghitungan	Pengecekan konsistensi setiap *user interface*

Tabel 5.4 menunjukkan bahwa Operational Consistency dihitung dengan cara mencari fitur yang memiliki proses yang inkonsisten. Tabel 5.5 menunjukkan bahwa Message Clarity dihitung dengan cara memeriksa setiap pesan ke pengguna yang muncul pada modul Ekuivalensi. Tabel 5.6 menunjukkan bahwa Functional Customizability dihitung dengan cara mencari fitur yang dapat diatur oleh pengguna. Tabel 5.7 menunjukkan bahwa User Interface Customizability dihitung dengan cara mencari *user interface* pada modul Ekuivalensi yang dapat diatur oleh pengguna.

Tabel 5.8 menunjukkan bahwa Monitoring Capability dihitung dengan cara mencari fungsi pada modul Ekuivalensi yang dapat dimonitor. Tabel 5.9 Tabel menunjukkan bahwa Undo Capability dihitung dengan cara mencari fitur dapat dilakukan *undo* agar bisa kembali ke kondisi sebelumnya dengan cepat. Tabel 5.10 menunjukkan bahwa Terminology Understandability dihitung dengan cara memeriksa istilah-istilah yang dapat dimengerti pada *user interface* modul Ekuivalensi. Tabel 5.11 menunjukkan bahwa Appearance Consistency dihitung dengan cara memeriksa konsistensi setiap *user interface* pada modul Ekuivalensi.

5.1.2 Maintainability

Maintainability menilai sejauh mana seberapa efektif dan efisien suatu produk atau sistem dapat dimodifikasi oleh pengelola. Pada tugas akhir ini dipilih 4 sub-kualitas dari Maintainability yaitu Modularity, Reusability, Analysability, dan Testability.

5.1.2.1 Modularity

Modularity digunakan untuk mengukur sejauh mana sebuah perubahan sebuah komponen pada suatu sistem memiliki pengaruh minimal ke komponen lain. Modularity mempunyai 2 poin kualitas yaitu Coupling of Components Conformance dan Cyclomatic Complexity.

Tabel 5.12 Coupling of Components Conformance

Nama	Coupling of Components Conformance
ID	MMo-1-G
Deskripsi	Seberapa independen sebuah komponen dan berapa banyak komponen yang bebas dari efek perubahan komponen lain dalam sistem
Rumus	X = A / B
A	Jumlah komponen yang diimplementasikan dengan sedikit pengaruh dengan komponen lain
B	Jumlah komponen yang harus independen
Penghitungan	• Mengecek *source code class file* (selain Framework) • Komponen yang memiliki dampak minimum adalah kelas yang tidak memerlukan perubahan dari kelas lain ketika kelas tersebut diubah. • Komponen yang seharusnya independen adalah kelas yang berisi proses transaksional dengan basis data.

Tabel 5.13 Cyclomatic Complexity

Nama	Cyclomatic Complexity
ID	MMo-2-S
Deskripsi	Jumlah modul yang mempunyai Cyclomatic Complexity dengan jumlah yang dapat diterima
Rumus	X = 1 - A / B
A	Jumlah modul yang dengan Cyclomatic Complexity melebihi batas
B	Jumlah seluruh modul
Penghitungan	Menghitung Cyclomatic Complexity pada setiap *method* pada *source code class file* (selain Framework). Penghitungan dilakukan dengan bantuan aplikasi Cyvis [13] dengan *threshold* yaitu 10

Tabel 5.12 menunjukkan bahwa Coupling of Components Conformance dihitung dengan cara menghitung tingkat pengaruh sebuah komponen ke komponen lain. Kriteria pertama dari komponen yang memiliki pengaruh minimum adalah komponen

yang tidak memerlukan perubahan kelas lain untuk melakukan perubahan. Komponen yang berisi transaksi ke basis data juga dapat dikategorikan mempunyai pengaruh minimal, namun tetap harus melihat kriteria pertama. Tabel 5.13 menunjukkan bahwa Cyclomatic Complexity mengukur kompleksitas seluruh *method* yang ada pada *class*. Penghitungan Cyclomatic Complexity dilakukan dengan bantuan *tools* yaitu Cyvis.

5.1.2.2 Reusability

Reusablity menilai sejauh mana sebuah aset dalam sistem dapat digunakan oleh lebih dari satu sistem atau dapat digunakan untuk membangun aset lain. Sub-kualitas Reusablity memiliki 2 poin kualitas yaitu Reusablity of Assets dan Conformance to Coding Rules.

Tabel 5.14 Reusability of Assets

Nama	Reusability of Assets
ID	MRe-1-G
Deskripsi	Jumlah aset yang bisa digunakan ulang
Rumus	X = A / B
A	Jumlah aset yang didesain dan diimplementasikan untuk dapat digunakan ulang
B	Jumlah seluruh aset
Penghitungan	• Mengecek *source code class file* (selain Framework) • Komponen yang dibuat untuk dapat digunakan ulang adalah kelas yang berisi proses lebih dari satu fitur atau tidak dibuat spesifik untuk membantu satu kelas tertentu.

Tabel 5.15 Conformance to Coding Rules

Nama	Conformance to Coding Rules
ID	MRe-2-S
Deskripsi	Jumlah modul yang dikembangkan sesuai *coding rules*

Rumus	X = A / B
A	Jumlah modul yang sesuai dengan *coding rules*
B	Jumlah modul dalam sistem
Penghitungan	• Mengecek *source code class file* (selain Framework) • Pengecekan dilakukan dengan bantuan sebuah *tools* yaitu Checkstyle, sebuah *plugin* dari Eclipse

Tabel 5.14 menunjukkan bahwa Reusability of Assets dihitung dengan cara mencari aset yang dapat digunakan ulang. Kriteria aset yang dapat digunakan ulang yaitu berisi proses lebih dari satu fitur atau tidak dibuat spesifik untuk membantu *class* tertentu. Tabel 5.15 menunjukkan bahwa Conformance to Coding Rules menilai apakah suatu *class* telah memenuhi *coding rules*. Penilaian *coding rules* dibantu dengan *tools* yaitu Checkstyle yang merupakan sebuah *plugin* dari Eclipse IDE.

5.1.2.3 Analysability

Analysability mengukur apakah memungkinkan untuk menilai suatu dampak perubahan pada produk atau sistem, atau untuk mendiagnosis kekurangan dan kegagalan produk yang dapat terjadi, atau untuk mengidentifikasi bagian yang akan dimodifikasi. Analysability memiliki 3 poin kualitas yaitu System Log Completeness Conformance, Diagnosis Function Effectiveness, dan Diagnosis Function Sufficiency Conformance.

Tabel 5.16 System Log Completeness Conformance

Nama	System Log Completeness Conformance
ID	Man-1-G
Deskripsi	Seberapa jauh jangkauan log sistem dalam mencatat operasi-operasi dalam sistem
Rumus	X = A / B
A	Jumlah log yang direkam di dalam sistem
B	Jumlah log dimana jejak prosesnya dibutuhkan sistem

Penghitungan	Mengecek adanya log di setiap fitur

Tabel 5.17 Diagnosis Function Effectiveness

Nama	Diagnosis Function Effectiveness
ID	Man-2-S
Deskripsi	Proporsi sejauh mana implementasi fitur yang sesuai dengan kebutuhan dibandingkan dengan fitur yang telah diimplementasikan
Rumus	X = A / B
A	Jumlah fitur yang menjawab kebutuhan
B	Jumlah fitur yang ada pada program
Penghitungan	Mengecek proses utama dan dokumentasi

Tabel 5.18 Diagnosis Function Sufficiency Conformance

Nama	Diagnosis Function Sufficiency Conformance
ID	Man-3-S
Deskripsi	Mengukur sejauh mana fitur-fitur memenuhi spesifikasi kebutuhan
Rumus	X = A / B
A	Jumlah fitur yang telah dibuat berdasarkan dokumentasi
B	Jumlah fitur yang seharusnya dibuat sesuai dengan dokumentasi
Penghitungan	Mengecek proses utama dan dokumentasi

Tabel 5.16 menunjukkan bahwa System Log Completeness Conformance dihitung dengan cara memeriksa jumlah log yang ada pada modul Ekuivalensi. Tabel 5.17 menunjukkan bahwa Diagnosis Function Effectiveness dihitung dengan cara menghitung implementasi fitur yang sudah sesuai kebutuhan. Tabel 5.18 menunjukkan bahwa Diagnosis Function Sufficiency Conformance dihitung dengan cara melihat jumlah fitur apakah sudah sesuai dengan dokumentasi. Penghitungan Diagnosis Function Effectiveness dan Diagnosis Function Sufficiency Conformance dibantu dengan melihat dokumentasi Buku Tugas Akhir [6].

5.1.2.4 Testability

Testability mengukur sejauh mana kriteria sebuah tes dapat diterapkan ke sebuah sistem dan apakah beberapa tes bisa dilakukan untuk memenuhi kriteria-kriteria tersebut. Testability memiliki 3 poin kualitas yaitu Test Function Completeness Conformance, Autonomous Testability, dan Test Restartability.

Tabel 5.19 Test Function Completeness Conformance

Nama	Test Function Completeness Conformance
ID	MTe-1-G
Deskripsi	Seberapa lengkap uji coba terhadap sistem
Rumus	X = A / B
A	Jumlah uji coba yang dilakukan
B	Jumlah uji coba yang seharusnya dilakukan
Penghitungan	Melihat dokumentasi Buku Tugas Akhir [6]

Tabel 5.20 Autonomous Testability

Nama	Autonomous Testability
ID	MTe-2-S
Deskripsi	Seberapa kemampuan uji coba secara *autonomous*
Rumus	X = A / B
A	Jumlah *stub* yang dapat berjalan pada *dependency test*
B	Jumlah uji coba yang harusnya dilakukan
Penghitungan	Melihat dokumentasi Buku Tugas Akhir [6]

Tabel 5.21 Test Restartability

Nama	Test Restartability
ID	MTe-3-S
Deskripsi	Seberapa mudah uji coba ulang setelah melakukan perbaikan sistem
Rumus	X = A / B
A	Jumlah uji coba yang memiliki titik *pause* dan *restart*
B	Jumlah uji coba yang memiliki titik *pause*
Penghitungan	Melihat dokumentasi Buku Tugas Akhir [6]

Tabel 5.19 menunjukkan bahwa Test Function Completeness Conformance dihitung dengan cara menghitung jumlah uji coba yang dilakukan apakah sudah sesuai dengan jumlah tes yang seharusnya dilakukan. Tabel 5.20 menunjukkan bahwa Autonomous Testability dihitung dengan cara menghitung jumlah *stub* yang dapat berjalan pada *dependency test*. Tabel 5.21 menunjukkan bahwa Test Restartability dihitung dengan cara menghitung jumlah tes yang dapat dilakukan *pause* dan *restart*. Seluruh penghitungan dari poin kualitas pada Testability dilakukan dengan melihat dokumentasi Buku Tugas Akhir [6].

5.2 Hasil Pengukuran Kualitas Program Eksisting

Pada hasil pengukuran kualitas program eksisting akan ditunjukkan parameter fungsi dari masing-masing poin kualitas yang telah ditentukan dalam ISO 25023. Parameter-parameter tersebut yaitu jumlah variabel A, jumlah variabel B, rumus yang digunakan, dan hasil dari jumlah variabel A dan B yang telah dihitung berdasarkan rumus. Hasil pengukuran kualitas akan selalu tepat ataupun berada diantara 0 dan 1. Jika semakin mendekati 0 atau bernilai 0 maka semakin buruk kualitasnya. Jika semakin mendekati 1 atau bernilai 1 maka kualitasnya semakin baik.

5.2.1 Usability

Tabel 5.22 Appropriateness Recognisability

Poin Kualitas	A	B	Rumus	Hasil
Description Completeness	11	11	X = A/B	1
Demonstration Capability	28	29	X = A/B	0,966
Entry Point Self-descriptiveness	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Appropriateness Recognisability di Tabel 5.22 adalah sebagai berikut:

- Description Completeness bernilai 1 karena dokumentasi berupa Buku Tugas Akhir [6] sudah menjelaskan langkah-langkah yang ada dalam setiap fitur di modul Ekuivalensi.
- Demonstration Capability tidak bernilai sempurna karena salah satu proses pada fitur yaitu Pengelolaan Ekuivalensi Peserta Didik Alih Jenjang kurang menunjukkan apa yang dilakukan pada halaman fiturnya.
- Entry Point Self-descriptiveness bernilai 1 karena setiap halaman utama pada fitur sudah dapat menjelaskan fitur yang diimplementasikan oleh halaman tersebut.

Tabel 5.23 Operability

Poin Kualitas	A	B	Rumus	Hasil
Operational Consistency	0	11	X = 1 - A/B	1
Message Clarity	47	47	X = A/B	1
Functional Customizability	0	11	X = A/B	0
User Interface Customizability	0	11	X = A/B	0
Monitoring Capability	0	11	X = A/B	0
Undo Capability	0	11	X = A/B	0
Terminology Understandability	29	29	X = A/B	1
Appearance Consistency	0	29	X = 1 - A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Operability di Tabel 5.23 adalah sebagai berikut:

- Operational Consistency bernilai 1 karena setiap proses pada fitur utama bersifat konsisten. Tidak ada perbedaan cara memasukan data, melihat data, menghapus data, dan proses-proses lainnya.
- Message Clarity bernilai 1 karena setiap pesan berupa *pop-up* yang ditampilkan selalu sama dan konsisten untuk setiap proses pada fitur utama.

- Functional Customizability bernilai 0 karena setiap proses dalam fitur utama tidak bisa diubah-ubah oleh pengguna.
- User Interface Customizability bernilai 0 karena tampilan *user interface* setiap proses pada fitur utama tidak bisa diubah-ubah oleh pengguna.
- Monitoring Capability bernilai 0 karena sama sekali tidak ada pencatatan tentang aktivitas pengguna selama menjalankan fitur utama.
- Undo Capability bernilai 0 karena tidak ada proses dalam fitur utama yang dapat melakukan *undo*.
- Terminology Understandability bernilai 1 karena setiap tampilan dapat dimengerti oleh pengguna yang dituju. Kesimpulan didapat setelah melihat jawaban kuesioner yang diisi oleh klien pada dokumentasi Buku Tugas Akhir yang menjawab "cukup" dan "sudah dimengerti".
- Appearance Consistency bernilai 1 karena semua tampilan sangat konsisten contohnya bagaimana menampilkan data dalam bentuk tabel, warna yang digunakan pada elemen-elemen *user interface*, posisi *button*, dan lain sebagainya.

5.2.2 Maintainability

Tabel 5.24 Modularity

Poin Kualitas	A	B	Rumus	Hasil
Coupling of Components Conformance	99	99	X = A/B	1
Cyclomatic Complexity	23	1443	X = 1 - A/B	0.984

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Modularity di Tabel 5.24 adalah sebagai berikut:

- Coupling of Components Conformance bernilai 1 karena *class* yang bersifat independen dengan yang seharusnya bersifat independen berjumlah sama. *Class* yang independen adalah Controller, Model, Domain, dan Interface. Sementara semua implementasi dari Interface

memang tidak seharusnya independen sehingga keluar dari hitungan.

- Cyclomatic Complexity bernilai tidak sempurna karena ada 23 dari 1443 *method* yang nilainya melebihi *threshold* yaitu 10.

Tabel 5.25 Reusability

Poin Kualitas	A	B	Rumus	Hasil
Reusability of Assets	99	153	X = A/B	0.647
Conformance to Coding Rules	0	153	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Reusability di Tabel 5.25 adalah sebagai berikut:

- Reusability of Assets bernilai tidak sempurna karena hanya adanya implementasi dari Interface dimana setiap fitur dari implementasi dibuat khusus untuk Interface.
- Conformance to Coding Rules bernilai 0 karena seluruh *class* dideteksi oleh Checkstyle mengandung *code violation* dimana paling banyak adalah masalah indentasi.

Tabel 5.26 Analysability

Poin Kualitas	A	B	Rumus	Hasil
System Log Completeness Conformance	0	29	X = A/B	0
Diagnosis Function Effectiveness	11	11	X = A/B	1
Diagnosis Function Sufficiency Conformance	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Analysability di Tabel 5.26 adalah sebagai berikut:

- System Log Completeness Conformance bernilai 0 karena sama sekali tidak ada log untuk pencatatan pada modul Ekuivalensi.
- Diagnosis Function Effectiveness bernilai 1 karena seluruh fitur menjawab kebutuhan yang ada.

- Diagnosis Function Sufficiency Conformance bernilai 1 karena semua fitur yang ada pada dokumentasi telah diimplementasikan.

Tabel 5.27 Testability

Poin Kualitas	A	B	Rumus	Hasil
Test Function Completeness Conformance	34	34	X = A/B	1
Autonomous Testability	0	34	X = A/B	0
Test Restartability	0	34	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Testability di Tabel 5.27 adalah sebagai berikut:

- Test Function Completeness Conformance bernilai 1 karena setiap fitur utama telah dites sesuai dengan dokumentasi Buku Tugas Akhir [6].
- Autonomous Testability bernilai 0 karena tidak ada catatan apapun mengenai tes yang menggunakan *stub* dalam dokumentasi Buku Tugas Akhir [6].
- Test Restartability bernilai 0 karena tidak ada catatan apapun mengenai tes dengan *pause* dan *restart* pada dokumentasi Buku Tugas Akhir [6].

5.3 Hasil Pengukuran Kualitas Pola Perancangan Alternatif

Tabel 5.28 Hasil pengukuran 3 pola perancangan

Poin Kualitas	Pola Perancangan		
	Program Eksisting	Active Record	Data Mapper
Description Completeness	1	1	1
Demonstration Capability	0,966	0,966	0,966
Entry Point Self-descriptiveness	1	1	1
Operational Consistency	1	1	1
Message Clarity	1	1	1
Functional Customizability	0	0	0

Poin Kualitas	Pola Perancangan		
	Program Eksisting	Active Record	Data Mapper
User Interface Customizability	0	0	0
Monitoring Capability	0	0	0
Undo Capability	0	0	0
Terminology Understandability	1	1	1
Appearance Consistency	1	1	1
Coupling of Components Conformance	1	1	1
Cyclomatic Complexity	0.984	0.971	0.972
Reusability of Assets	0.647	1	1
Conformance to Coding Rules	0	0	0
System Log Completeness Conformance	0	0	0
Diagnosis Function Effectiveness	1	1	1
Diagnosis Function Sufficiency Conformance	1	1	1
Test Function Completeness Conformance	1	1	1
Autonomous Testability	0	0	0
Test Restartability	0	0	0

Tabel 5.28 menunjukkan hasil pengukuran kualitas dari 3 pola perancangan berdasarkan seluruh poin kualitas yang telah dipilih. Hasil pengukuran kualitas selalu tepat ataupun berada diantara 0 dan 1. Jika semakin mendekati 0 atau bernilai 0 maka semakin buruk kualitasnya. Jika semakin mendekati 1 atau bernilai 1 maka kualitasnya semakin baik. Tabel 5.28 juga menunjukkan bahwa hanya 2 poin kualitas yang berubah dibandingkan dengan kualitas program eksisting, yaitu Cyclomatic Complexity dari sub-kualitas Modularity dan Reusability of Assets dari sub-kualitas Reusability. Sementara poin kualitas lain tidak mengalami perubahan.

5.3.1 Pola Perancangan Active Record

Sama seperti pengukuran kualitas pada program eksisting, pada hasil pengukuran kualitas pola perancangan Active Record juga akan ditunjukkan parameter fungsi dari masing-masing poin kualitas yang telah ditentukan dalam ISO 25023. Parameter-parameter tersebut yaitu jumlah variabel A, jumlah variabel B, rumus yang digunakan, dan hasil dari jumlah variabel A dan B yang telah dihitung berdasarkan rumus. Hasil pengukuran kualitas akan selalu tepat ataupun berada diantara 0 dan 1. Jika semakin mendekati 0 atau bernilai 0 maka semakin buruk kualitasnya. Jika semakin mendekati 1 atau bernilai 1 maka kualitasnya semakin baik.

5.3.1.1 Usability

Pada parameter kualitas Usability, hasil pengukuran kualitas pada semua poin kualitas setelah perubahan pola perancangan ke Active Record sama sekali tidak ada perubahan. Hal ini disebabkan oleh perubahan pola perancangan ke Active Record yang ternyata hanya menyentuh *class source code*. Sementara fungsionalitas, fitur, proses bisnis, dan *user interface* sama sekali tidak berubah.

Tabel 5.29 Appropriateness Recognisability pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
Description Completeness	11	11	X = A/B	1
Demonstration Capability	28	29	X = A/B	0,966
Entry Point Self-descriptiveness	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Appropriateness Recognisability di Tabel 5.29 adalah sebagai berikut:

- Description Completeness bernilai 1 karena dokumentasi berupa Buku Tugas Akhir [6] sudah menjelaskan langkah-langkah yang ada dalam setiap fitur di modul Ekuivalensi.

- Demonstration Capability tidak bernilai sempurna karena salah satu proses pada fitur yaitu Pengelolaan Ekuivalensi Peserta Didik Alih Jenjang kurang menunjukkan apa yang dilakukan pada halaman fiturnya sehingga mengurangi nilai kemampuan demonstrasi.
- Entry Point Self-descriptiveness bernilai 1 karena setiap halaman utama pada fitur sudah dapat menjelaskan fitur yang diimplementasikan oleh halaman tersebut.

Tabel 5.30 Operability pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
Operational Consistency	0	11	X = 1 - A/B	1
Message Clarity	47	47	X = A/B	1
Functional Customizability	0	11	X = A/B	0
User Interface Customizability	0	11	X = A/B	0
Monitoring Capability	0	11	X = A/B	0
Undo Capability	0	11	X = A/B	0
Terminology Understandability	29	29	X = A/B	1
Appearance Consistency	0	29	X = 1 - A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Operability di Tabel 5.30 adalah sebagai berikut:

- Operational Consistency bernilai 1 karena setiap proses pada fitur utama bersifat konsisten. Tidak ada perbedaan cara memasukan data, melihat data, menghapus data, dan proses-proses lainnya.
- Message Clarity bernilai 1 karena setiap pesan berupa *pop-up* yang ditampilkan selalu sama dan konsisten untuk setiap proses pada fitur utama.
- Functional Customizability bernilai 0 karena setiap proses dalam fitur utama tidak bisa diubah-ubah oleh pengguna.

- User Interface Customizability bernilai 0 karena tampilan *user interface* setiap proses pada fitur utama tidak bisa diubah-ubah oleh pengguna.
- Monitoring Capability bernilai 0 karena sama sekali tidak ada pencatatan tentang aktivitas pengguna selama menjalankan fitur utama.
- Undo Capability bernilai 0 karena tidak ada proses dalam fitur utama yang dapat melakukan *undo*.
- Terminology Understandability bernilai 1 karena setiap tampilan dapat dimengerti oleh pengguna yang dituju. Kesimpulan didapat setelah melihat jawaban kuesioner yang diisi oleh klien pada dokumentasi Buku Tugas Akhir yang menjawab "cukup" dan "sudah dimengerti".
- Appearance Consistency bernilai 1 karena semua tampilan sangat konsisten contohnya bagaimana menampilkan data dalam bentuk tabel, warna yang digunakan pada elemen-elemen *user interface*, posisi *button*, dan lain sebagainya.

5.3.1.2 Maintainability

Pada parameter kualitas Maintainability, hanya 2 poin kualitas yang berubah dibandingkan dengan kualitas program eksisting, yaitu Cyclomatic Complexity dari sub-kualitas Modularity dan Reusability of Assets dari sub-kualitas Reusability.

Tabel 5.31 Modularity pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
Coupling of Components Conformance	44	44	X = A/B	1
Cyclomatic Complexity	22	756	X = 1 - A/B	0.971

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Modularity di Tabel 5.31 adalah sebagai berikut:

- Coupling of Components Conformance bernilai 1 dan tidak mengalami perubahan dari kualitas program eksisting karena jumlah *class* yang independen dengan

jumlah *class* yang seharusnya independen berjumlah sama. *Class* yang independen adalah Controller dan *class* Active Record.

- Cyclomatic Complexity menurun karena meski jumlah *method* yang melebihi *threshold* mengalami penurunan, namun jumlah keseluruhan *method* yang jauh lebih sedikit menyebabkan hasil penghitungan kualitas Cyclomatic Complexity menurun dari program eksisting.

Tabel 5.32 Reusability pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
Reusability of Assets	44	44	X = A/B	1
Conformance to Coding Rules	0	44	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Reusability di Tabel 5.32 adalah sebagai berikut:

- Reusability of Assets mengalami peningkatan dari program eksisting dan bernilai sempurna karena tidak diimplementasikannya *class* Interface sehingga seluruh *class* memenuhi kriteria yaitu tidak dibuat khusus untuk membantu *class* lain.
- Conformance to Coding Rules bernilai 0 karena seluruh *class* dideteksi oleh Checkstyle mengandung *code violation* dimana paling banyak adalah masalah indentasi.

Tabel 5.33 Analysability pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
System Log Completeness Conformance	0	29	X = A/B	0
Diagnosis Function Effectiveness	11	11	X = A/B	1
Diagnosis Function Sufficiency Conformance	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Analysability di Tabel 5.33 adalah sebagai berikut:

- System Log Completeness Conformance bernilai 0 karena sama sekali tidak ada log untuk pencatatan pada modul Ekuivalensi setelah perubahan ke Active Record.
- Diagnosis Function Effectiveness bernilai 1 karena seluruh fitur menjawab kebutuhan yang ada.
- Diagnosis Function Sufficiency Conformance bernilai 1 karena semua fitur yang ada pada dokumentasi telah diimplementasikan.

Tabel 5.34 Testability pada Active Record

Poin Kualitas	A	B	Rumus	Hasil
Test Function Completeness Conformance	34	34	X = A/B	1
Autonomous Testability	0	34	X = A/B	0
Test Restartability	0	34	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Testability di Tabel 5.34 adalah sebagai berikut:

- Test Function Completeness Conformance bernilai 1 karena setiap fitur utama telah dites sesuai dengan dokumentasi Buku Tugas Akhir [6].
- Autonomous Testability bernilai 0 karena tidak ada catatan apapun mengenai tes yang menggunakan *stub* dalam dokumentasi Buku Tugas Akhir [6].
- Test Restartability bernilai 0 karena tidak ada catatan apapun mengenai tes dengan *pause* dan *restart* pada dokumentasi Buku Tugas Akhir [6].

5.3.2 Pola Perancangan Data Mapper

Sama seperti pengukuran kualitas pada program eksisting dan Active Record, pada hasil pengukuran kualitas pola perancangan Data Mapper juga akan ditunjukkan parameter fungsi

dari masing-masing poin kualitas yang telah ditentukan dalam ISO 25023. Hasil pengukuran kualitas akan selalu tepat ataupun berada diantara 0 dan 1. Jika semakin mendekati 0 atau bernilai 0 maka semakin buruk kualitasnya. Jika semakin mendekati 1 atau bernilai 1 maka kualitasnya semakin baik.

5.3.2.1 Usability

Sama seperti Active Record, pada parameter kualitas Usability, hasil pengukuran kualitas pada semua poin kualitas setelah perubahan pola perancangan ke Data Mapper sama sekali tidak ada perubahan. Hal ini disebabkan oleh perubahan pola perancangan ke Data Mapper yang ternyata juga hanya menyentuh *class source code*. Sementara fungsionalitas, fitur, proses bisnis, dan *user interface* sama sekali tidak berubah.

Tabel 5.35 Appropriateness Recognisability pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
Description Completeness	11	11	X = A/B	1
Demonstration Capability	28	29	X = A/B	0,966
Entry Point Self-descriptiveness	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Appropriateness Recognisability Tabel 5.35 di adalah sebagai berikut:

- Description Completeness bernilai 1 karena dokumentasi berupa Buku Tugas Akhir [6] sudah menjelaskan langkah-langkah yang ada dalam setiap fitur di modul Ekuivalensi.
- Demonstration Capability tidak bernilai sempurna karena salah satu proses pada fitur yaitu Pengelolaan Ekuivalensi Peserta Didik Alih Jenjang kurang menunjukkan apa yang dilakukan pada halaman fiturnya sehingga mengurangi nilai kemampuan demonstrasi.

- Entry Point Self-descriptiveness bernilai 1 karena setiap halaman utama pada fitur sudah dapat menjelaskan fitur yang diimplementasikan oleh halaman tersebut.

Tabel 5.36 Operability pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
Operational Consistency	0	11	X = 1 - A/B	1
Message Clarity	47	47	X = A/B	1
Functional Customizability	0	11	X = A/B	0
User Interface Customizability	0	11	X = A/B	0
Monitoring Capability	0	11	X = A/B	0
Undo Capability	0	11	X = A/B	0
Terminology Understandability	29	29	X = A/B	1
Appearance Consistency	0	29	X = 1 - A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Operability di Tabel 5.36 adalah sebagai berikut:

- Operational Consistency bernilai 1 karena setiap proses pada fitur utama bersifat konsisten. Tidak ada perbedaan cara memasukan data, melihat data, menghapus data, dan proses-proses lainnya.
- Message Clarity bernilai 1 karena setiap pesan berupa *pop-up* yang ditampilkan selalu sama dan konsisten untuk setiap proses pada fitur utama.
- Functional Customizability bernilai 0 karena setiap proses dalam fitur utama tidak bisa diubah-ubah oleh pengguna.
- User Interface Customizability bernilai 0 karena tampilan *user interface* setiap proses pada fitur utama tidak bisa diubah-ubah oleh pengguna.

- Monitoring Capability bernilai 0 karena sama sekali tidak ada pencatatan tentang aktivitas pengguna selama menjalankan fitur utama.
- Undo Capability bernilai 0 karena tidak ada proses dalam fitur utama yang dapat melakukan *undo*.
- Terminology Understandability bernilai 1 karena setiap tampilan dapat dimengerti oleh pengguna yang dituju. Kesimpulan didapat setelah melihat jawaban kuesioner yang diisi oleh klien pada dokumentasi Buku Tugas Akhir yang menjawab "cukup" dan "sudah dimengerti".
- Appearance Consistency bernilai 1 karena semua tampilan sangat konsisten contohnya bagaimana menampilkan data dalam bentuk tabel, warna yang digunakan pada elemen-elemen *user interface*, posisi *button*, dan lain sebagainya.

5.3.2.2 Maintainability

Sama seperti Active Record, pada parameter kualitas Maintainability, hanya 2 poin kualitas yang berubah dibandingkan dengan kualitas program eksisting, yaitu Cyclomatic Complexity dari sub-kualitas Modularity dan Reusability of Assets dari sub-kualitas Reusability.

Tabel 5.37 Modularity pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
Coupling of Components Conformance	67	67	X = A/B	1
Cyclomatic Complexity	22	793	X = 1 - A/B	0.972

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Modularity di Tabel 5.37 adalah sebagai berikut:

- Coupling of Components Conformance bernilai 1 dan tidak mengalami perubahan dari kualitas program eksisting karena jumlah *class* yang independen dengan jumlah *class* yang seharusnya independen berjumlah sama.

Class yang independen adalah Controller, Mapper, dan Domain.

- Cyclomatic Complexity menurun karena meski jumlah *method* yang melebihi *threshold* mengalami penurunan, namun jumlah keseluruhan *method* yang jauh lebih sedikit menyebabkan hasil penghitungan kualitas Cyclomatic Complexity menurun dari program eksisting.

Tabel 5.38 Reusability pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
Reusability of Assets	67	67	X = A/B	1
Conformance to Coding Rules	0	67	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Reusability di Tabel 5.38 adalah sebagai berikut:

- Reusability of Assets mengalami peningkatan dari program eksisting dan bernilai sempurna karena tidak diimplementasikannya *class* Interface sehingga seluruh *class* memenuhi kriteria yaitu tidak dibuat khusus untuk membantu *class* lain.
- Conformance to Coding Rules bernilai 0 karena seluruh *class* dideteksi oleh Checkstyle mengandung *code violation* dimana paling banyak adalah masalah indentasi.

Tabel 5.39 Analysability pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
System Log Completeness Conformance	0	29	X = A/B	0
Diagnosis Function Effectiveness	11	11	X = A/B	1
Diagnosis Function Sufficiency Conformance	11	11	X = A/B	1

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Analysability di Tabel 5.39 adalah sebagai berikut:

- System Log Completeness Conformance bernilai 0 karena sama sekali tidak ada log untuk pencatatan pada modul Ekuivalensi setelah perubahan ke Data Mapper.
- Diagnosis Function Effectiveness bernilai 1 karena seluruh fitur menjawab kebutuhan yang ada.
- Diagnosis Function Sufficiency Conformance bernilai 1 karena semua fitur yang ada pada dokumentasi telah diimplementasikan.

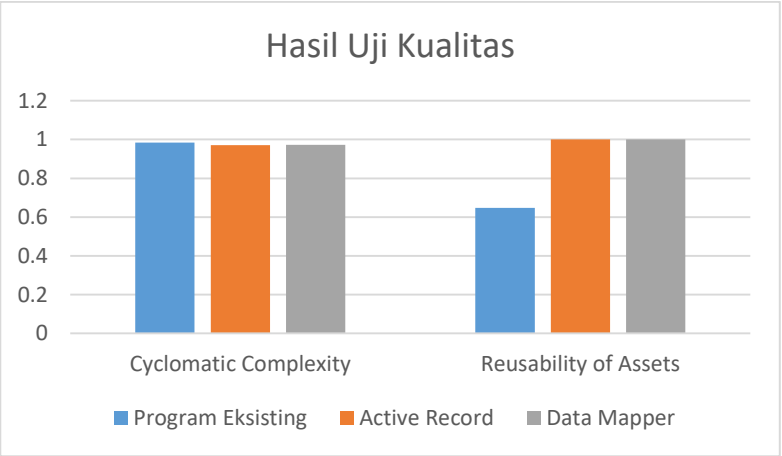
Tabel 5.40 Testability pada Data Mapper

Poin Kualitas	A	B	Rumus	Hasil
Test Function Completeness Conformance	34	34	X = A/B	1
Autonomous Testability	0	34	X = A/B	0
Test Restartability	0	34	X = A/B	0

Penjelasan dari hasil penghitungan kualitas pada sub-kualitas Testability di Tabel 5.40 adalah sebagai berikut:

- Test Function Completeness Conformance bernilai 1 karena setiap fitur utama telah dites sesuai dengan dokumentasi Buku Tugas Akhir [6].
- Autonomous Testability bernilai 0 karena tidak ada catatan apapun mengenai tes yang menggunakan *stub* dalam dokumentasi Buku Tugas Akhir [6].
- Test Restartability bernilai 0 karena tidak ada catatan apapun mengenai tes dengan *pause* dan *restart* pada dokumentasi Buku Tugas Akhir [6].

5.4 Evaluasi



Gambar 5.1 Perbandingan hasil uji kualitas

Gambar 5.1 menunjukkan bahwa perbandingan kualitas hanya dilakukan terhadap 2 poin kualitas yang mengalami perubahan yaitu Cyclomatic Complexity pada sub-kualitas Modularity dan Reusability of Assets pada sub-kualitas Reusability.

Pada poin kualitas Reusability of Assets, pola perancangan Active Record dan Data Mapper memiliki nilai lebih tinggi dibanding program eksisting. Hal ini terjadi karena Active Record dan Data Mapper tidak mengimplementasikan *class* Interface dan implementasinya sehingga setiap *class* tidak dibuat khusus untuk membantu *class* lain. Pada poin kualitas Cyclomatic Complexity, Active Record mempunyai nilai terendah karena meski jumlah *method* yang melebihi *threshold* mengalami penurunan, jumlah keseluruhan *method* jauh lebih sedikit sehingga menyebabkan Active Record memiliki hasil penghitungan kualitas Cyclomatic Complexity yang paling rendah.

Tabel 5.41 Rata-rata kualitas di setiap pola perancangan

Kode Poin Kualitas	Program Eksisting	Active Record	Data Mapper
Cyclomatic Complexity	0.984	0.971	0.972
Reusability of Assets	0.647	1	1
Rata-rata	0.816	0.9855	0.986

Tabel 5.41 menunjukkan bahwa Data Mapper memiliki kualitas yang paling baik secara keseluruhan dengan perbedaan yang sangat kecil dari Active Record. Ini disebabkan nilai Cyclomatic Complexity yang sedikit lebih tinggi dari Active Record. Selain itu Data Mapper memiliki nilai Reusability of Assets yang sama dengan Active Record dan keduanya lebih baik dari program eksisting dengan selisih hasil yang lebih tinggi dari selisih hasil antar Cyclomatic Complexity.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir beserta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perubahan pola perancangan dan pengukuran kualitas sebelum dan sesudah perubahan pola perancangan, dapat diambil kesimpulan sebagai berikut.

1. Pada modul Ekuivalensi, penerapan pola perancangan Active Record diterapkan dengan cara menyatukan fungsi-fungsi utama pada Service, Repository, dan Domain dari pola perancangan awal. Sementara itu pola perancangan Data Mapper diterapkan dengan cara menyatukan fungsi-fungsi utama Service dan Repository.
2. Kualitas dari Usability dan Maintainability berhasil diukur dengan cara melihat detil poin-poin kualitas yang dipilih dan menghitung hasil pengukurannya dengan menggunakan formula yang telah disediakan pada setiap poin-poin kualitas tersebut.
3. Hasil pengukuran kualitas modul Ekuivalensi berhasil dievaluasi dengan cara membandingkan naik dan turunnya hasil dari pengukuran setelah dihitung dengan formula yang disediakan di ISO 25023.
4. Perubahan pola perancangan mempengaruhi kualitas modul Ekuivalensi karena setelah penghitungan kualitas dari pola perancangan Active Record dan Data Mapper, terdapat perubahan hasil kualitas baik peningkatan ataupun penurunan.

6.2 Saran

Terdapat saran-saran terkait Tugas Akhir ini yang diharapkan dapat membuat tugas akhir ini menjadi lebih baik. Saran-saran tersebut yaitu:

1. Penambahan parameter kualitas selain Usability dan Maintainability. Pada tugas akhir ini perubahan pola perancangan ternyata kurang menyentuh poin kualitas terutama pada Usability. Hal ini dikarenakan Active Record dan Data Mapper tidak menyentuh fitur, *user interface*, dan proses bisnis.
2. Penambahan atau penggunaan pola perancangan lain karena Active Record dan Data Mapper yang merupakan pola perancangan *data source* kurang menyentuh poin-poin kualitas terutama pada Usability karena tidak ada penggantian fitur, *user interface*, dan proses bisnis.

DAFTAR PUSTAKA

[1] G. P. N. Suminto, U. L. Yuhana and R. N. E. Anggraini, Rancang Bangun Commercial Off The Self (COTS) Sistem Informasi Akademik Berbasis Web pada Modul Kelola Pembelajaran, Jur. Tek. Inform.-ITS, 2015.

[2] U. L. Yuhana, S. A. Wijaya and R. J. Akbar, Rancang Bangun Kerangka Kerja Sistem Informasi Akademik Modular Berbasis Web dengan Pola Arsitektur Hierarchical Model-View-Controller, Jur. Tek. Inform.-ITS, 2016.

[3] T. Nurwantoro, U. L. Yuhana and R. J. Akbar, Kerangka Kerja Sinkronisasi Basis Data Relasional Berbasis Web pada Studi Kasus Sistem Informasi Akademik, Jur. Tek. Inform.-ITS, 2015.

[4] H. Rahman, S. Rochimah and R. N. E. Anggraini, Rancang Bangun Sistem Informasi Akademik Generik pada Modul Penilaian Menggunakan Pola Perancangan Hierarchical Model-View-Controller, Jur. Tek. Inform.-ITS, 2015, 2015.

[5] A. T. Averousi, S. Rochimah and R. J. Akbar, Rancang Bangun Perangkat Lunak Sistem Informasi Akademik Generik pada Modul Kurikulum, Jur. Tek. Inform.-ITS, 2015.

[6] B. A. Alfirdaus, U. L. Yuhana and R. N. E. Anggraini, Rancang Bangun Perangkat Lunak Sistem Informasi Akademik Berbasis Web dengan Rancangan Modularitas dan Evolusi pada Modul Ekuivalensi, Jur. Tek. Inform.-ITS, 2015.

[7] D. R. H, Software Quality: Concepts and Plans, First, New Jersey: Prentice Hall, 1989.

[8] I. R. Eko, Kriteria Penjaminan Kualitas Perangkat Lunak, Seri 999 E-Artik, 2012.

[9] "ISO 25010," [Online]. Available: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Accessed 19 12 2016].

[10] "ISO/IEC 25023:2016(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality," [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25023:ed-1:v1:en. [Accessed 19 12 2016].

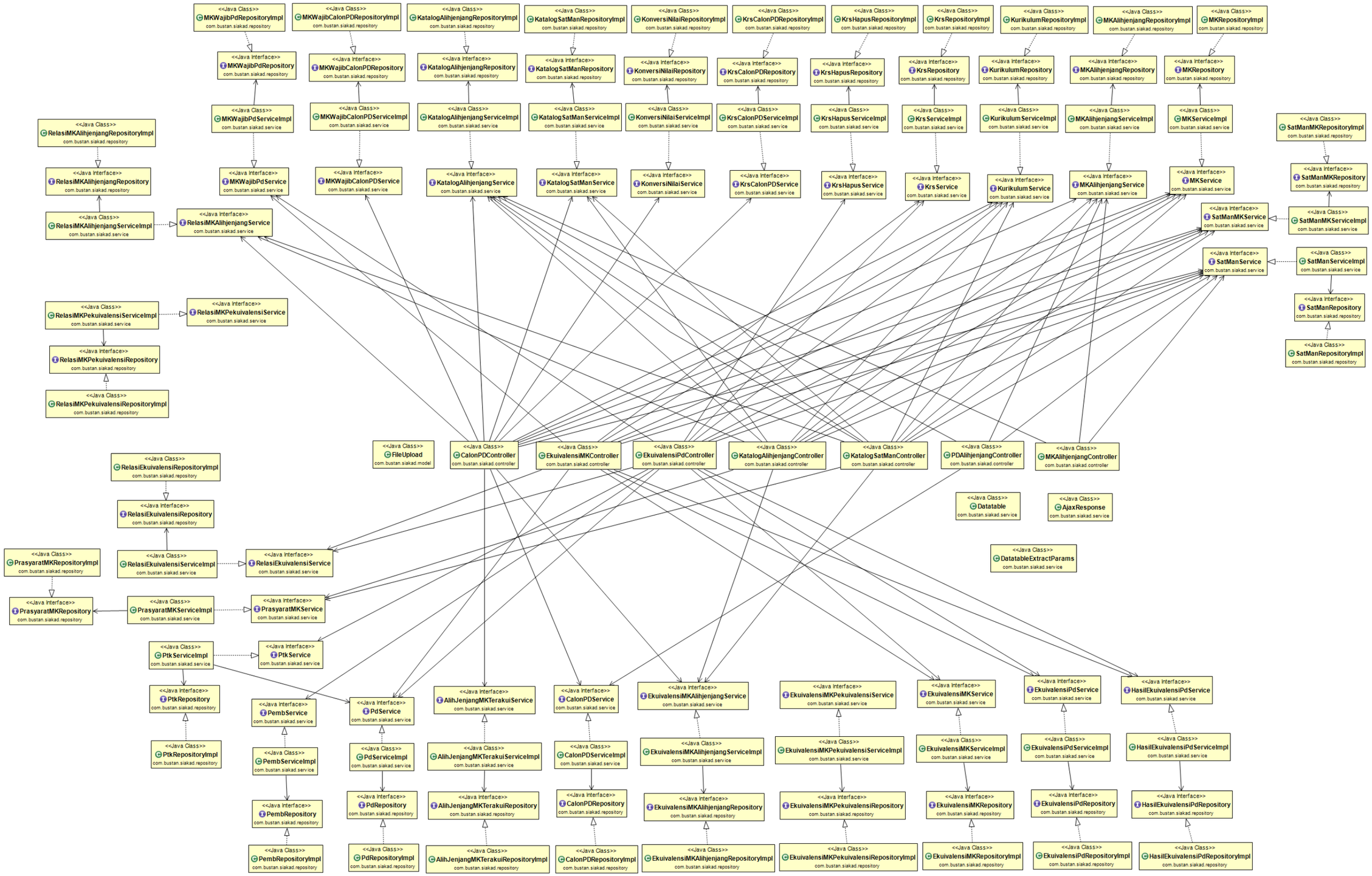
[11] S. Board, IEEE Standard Glossary of Software Engineering Terminology, New York: The Institute of Electrical and Electronics Ehgineers, 1990.

[12] M. Fowler, Patterns of Enterprise Application Architecture, USA: Addison-Wesley Professional, 2002.

[13] Richard, "Richard JP Le Guen.ca," [Online]. Available: http://richard.jp.leguen.ca/tutoring/soen343-f2010/tutorials/implementing-data-mapper/. [Accessed 10 June 2017].

[14] P. Selvaraj and V. I. , "Cyvis, Software Complexity Visualiser," 2006. [Online]. Available: http://cyvis.sourceforge.net/. [Accessed 4 June 2017].

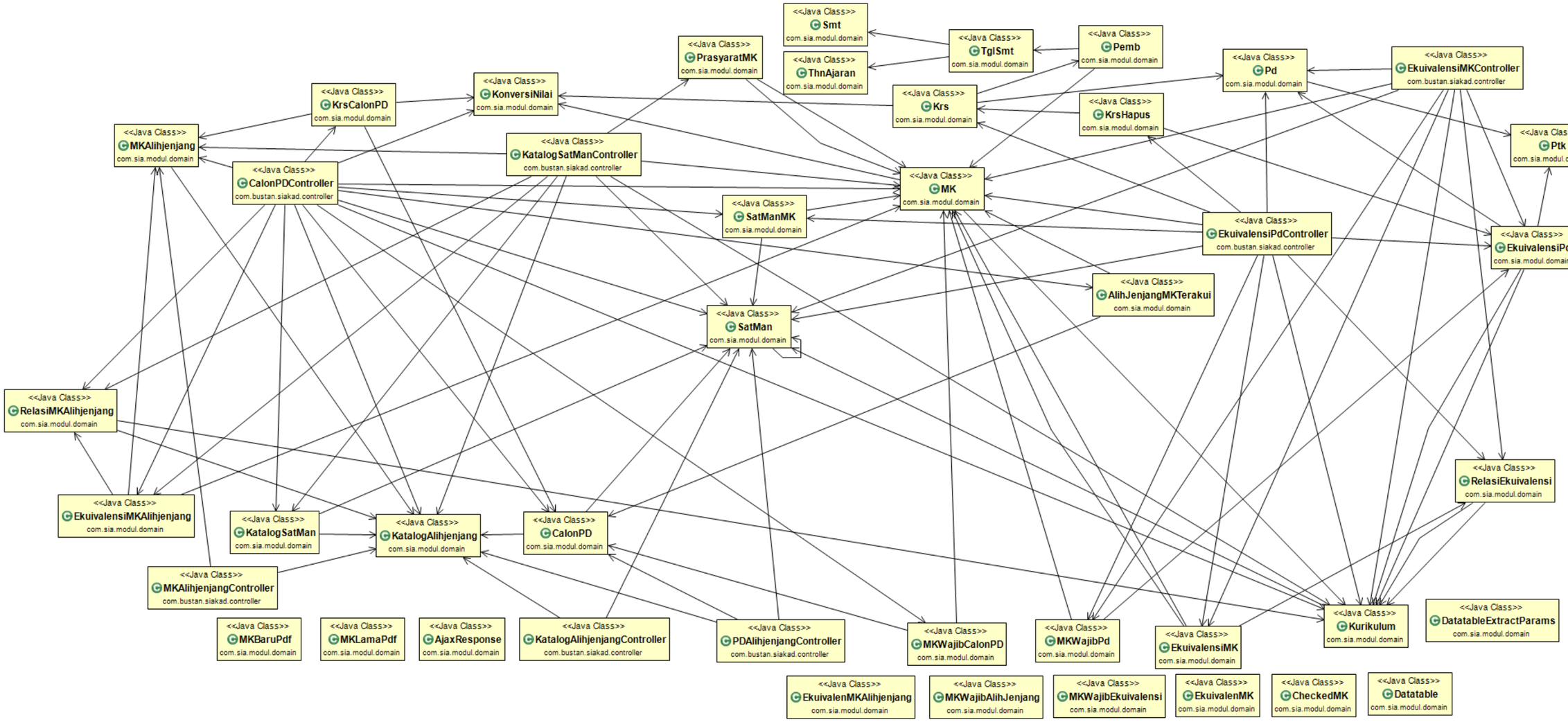
LAMPIRAN A. DIAGRAM KELAS PROGRAM EKSISTING



Lampiran A Diagram kelas program eksistin

[Halaman ini sengaja dikosongkan]

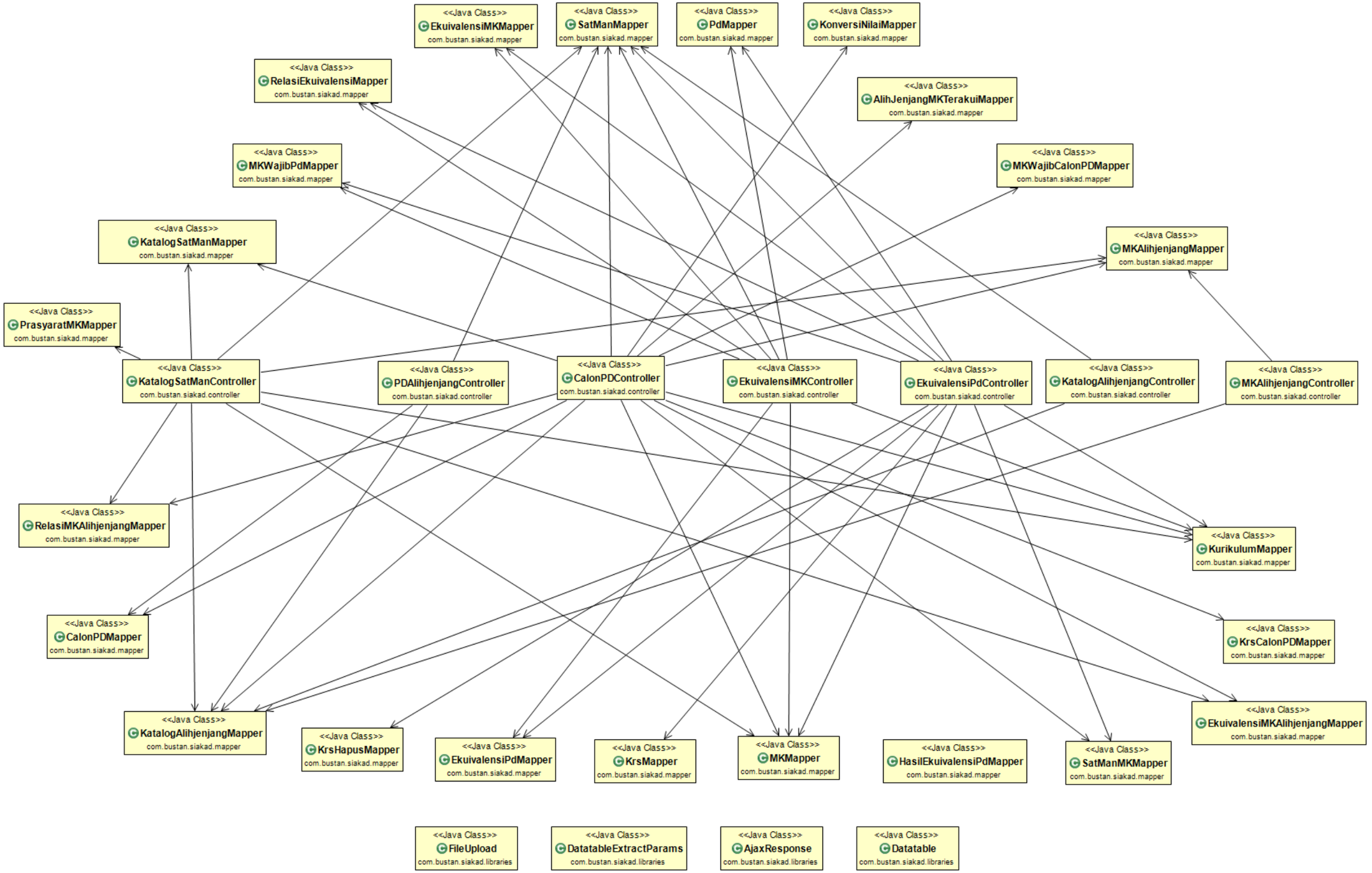
LAMPIRAN B. DIAGRAM KELAS ACTIVE RECORD



Lampiran B Diagram kelas Active Record

[Halaman ini sengaja dikosongkan]

LAMPIRAN C. DIAGRAM KELAS DATA MAPPER



Lampiran C Diagram kelas Data Mapper

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis, Afif Ishamsyah Hantriono lahir di Magetan pada tanggal 13 September 1995 dan dibesarkan di Kota Bekasi. Penulis adalah anak pertama dari dua bersaudara.

Penulis telah menempuh pendidikan formal pada jenjang TK sampai dengan S-1 di TK Islamic Izhar (1999-2001), SDIT An-Nadwah Tambun Selatan (2001-2007), SMP Al-Azhar Syifa Budi Legenda Bekasi (2007-2010), SMA Al-Muslim Tambun (2010-2013), Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya (2013-2017).

Di Jurusan Teknik Informatika, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL). Penulis juga aktif dalam organisasi kemahasiswaan mulai dari HMTC (Himpunan Mahasiswa Teknik Computer-Informatika) sebagai Staf Departemen Dalam Negeri (2014-2015). Penulis pernah menjadi anggota Keamanan dan Transportasi Schematics 2014 dan menjadi Koordinator Keamanan dan Perizinan Schematics 2015. Penulis dapat dihubungi pada alamat email afifhan95@gmail.com