PEMBUATAN MODEL *SEQUENCE DIAGRAM* DENGAN *REVERSE ENGINEERING* APLIKASI BASIS DATA PADA *SMARTPHONE* UNTUK MENJAGA KONSISTENSI DESAIN PERANGKAT LUNAK

Indra Kharisma Raharjana [1)], Army Justitia [2)]
[1, 2)] Program Studi Sistem Informasi, Universitas Airlangga
Kampus C Mulyorejo Surabaya
e-mail: indra.kharisma@fst.unair.ac.id[1)], army-j@fst.unair.ac.id[2)]

ABSTRAK

Konsistensi desain sequence diagram dengan implementasi sering menjadi masalah dalam pembangunan perangkat lunak, penelitian ini akan menerapkan hasil reverse engineering sequence diagram dari aplikasi yang telah ada sebagai model dasar untuk membangun perangkat lunak sehingga konsistensi desain sequence diagram bisa dijaga. Dalam penelitian ini model sequence diagram tersebut dibangun dengan cara mengklasifikasikan interaksi antara pengguna dengan perangkat lunak di aplikasi basis data pada smartphone. Untuk membuat model UML Sequence Diagram, langkah pertama adalah memilih dataset berupa source code aplikasi contoh di Blackberry 10 dan kemudian memahami fungsional aplikasi tersebut. Langkah selanjutnya adalah melakukan reverse engineering dari source code tersebut berdasarkan fitur yang telah didefinisikan sebelumnya. Lalu dilakukan pembangunan model UML sequence diagram berdasarkan kesamaan dan pengelompokan pola pada hasil reverse engineering. Langkah terakhir adalah melakukan evaluasi terhadap model UML sequence diagram dengan cara menggunakannya pada pembangunan aplikasi baru, mengukur tingkat konsistensi desain sequence diagram serta mengambil pengetahunan dari penerapan studi kasus. Berdasarkan hasil penelitian, model sequence diagram yang didapatkan dari proses reverse engineering bisa menjaga konsistensi antara desain sequence diagram dengan implementasi pada pembangunan perangkat lunak.

***Kata Kunci**: Sequence Diagram, Basis Data, Reverse Engineering, Smartphone*

ABSTRACT

Ensuring consistency between design and implementation of sequence diagrams in software development is often a problem in software development. This study will be applying reverse engineering from existing applications to make sequence diagrams as a model to build software so consistency of design can be maintained. To create UML sequence diagram model, the first step was selecting the dataset of sample source code in Blackberry 10 and then understood its functionality. Reverse engineering was performed from the source code based on the features that have been defined previously. UML sequence diagram model were constructed based on similarity and grouping patterns. The evaluation of UML Sequence Diagram model was performed by developing new Smartphone application, measuring the level of design consistency, and examined the case studies to take knowledge. Based on the research results, the model sequence diagrams obtained from reverse engineering process can maintain consistency between the design sequence diagrams and implementation of software development.

***Keywords**: Sequence Diagram, Database, Reverse Engineering, Smartphone*

I. PENDAHULUAN

UNIFIED Modelling Language (UML) adalah bahasa pemodelan visual yang digunakan untuk menentukan, memvisualisasi, membangun, serta mendokumentasikan perangkat lunak . UML memiliki lebih dari 14 jenis diagram yang secara umum dapat dikategorikan menjadi dua yaitu diagram yang menggambarkan informasi struktural dan diagram yang menjelaskan tentang informasi perilaku. Untuk merancang perangkat lunak disarankan hanya menggunakan diagram-diagram yang benar-benar diperlukan [1]. komponen UML yang paling sering digunakan menurut survey kepada profesional berpengalaman secara berutan adalah *class diagram, use case diagram, sequence diagram ,use case narative,activity diagram, statechart diagram,* dan *collaboration diagram* [2].

Pengembangan sistem perangkat lunak saat ini berkembang sangat pesat dan cepat berubah sehingga membuat pengembang merilis sistem perangkat lunak tanpa dokumentasi yang benar[3], sehingga permasalahan yang sering muncul dalam pengembangan perangkat lunak adalah konsistensi antara diagram UML dengan implementasi [4]. Dalam pembuatan perangkat lunak dibutuhkan pengalaman teknis untuk memastikan kesesuaian desain dengan keputusan arsitektur yang dibuat [5]. Biasanya diagram UML yang dibuat hanya memuat alur logika saja, padahal setiap bahasa pemrograman memiliki karakteristik implementasi yang berbeda. *Sequence diagram* yang merupakan komponen UML yang sering dipakai dalam pembangunan perangkat lunak

merupakan diagram yang beresiko terhadap konsistensi antara model dan implementasi karena diagram ini menggambarkan aktivitas dari interaksi antar objek *class* yang terjadi pada *use case scenario* [4][6-7].

Untuk menjaga konsistensi antara desain *sequence diagram* dan implentasi, dalam penelitian ini diusulkan untuk membuat pola interaksi menggunakan *sequence diagram* berdasarkan *reverse engineering*. Hipotesis dari penelitian ini adalah konsistensi antara desain dan implementasi bisa dijaga dengan pola *sequence diagram* yang didapatkan dari proses *reverse engineering*, sehingga pola *sequence diagram* ini bisa digunakan untuk melakukan desain aplikasi. Tahapan yang dilakukan dalam penelitian ini adalah mengumpulkan *source code* aplikasi sebagai data untuk melakukan *reverse engineering*, hasil *reverse engineering* tersebut adalah *sequence diagram* yang kemudian dikelompokan berdasarkan kesamaan fitur untuk dibuat pola dari *sequence diagram*. Pola dari *sequence diagram* tersebut kemudian diimplementasikan pada studi kasus untuk menguji konsistensi antara desain dengan implementasi. Karena bahasa pemrograman memiliki karakteristik yang berbeda-beda dalam penerapan desain serta implementasinya, maka dalam penelitian ini hanya akan fokus pada *platform* pengembangan aplikasi *smartphone* untuk Blackberry 10 yang menggunakan bahasa C++, pemilihan platform ini didasarkan pada tersedianya *dataset* contoh aplikasi yang memadai untuk penyusunan pola *sequence diagram* menggunakan *reverse engineering*.

Informasi yang berhubungan dengan struktur dan interaksi pada sistem dibutuhkan untuk memahani sistem perangkat lunak yang dikembangkan dengan *object-oriented*. Untuk memahami struktur dan pola interaksi yang ada digunakan metode *reverse engineering*. *Reverse engineering* adalah proses menganalisa sistem dengan maksud mengidentifikasi komponen-komponen sistem dan interaksi antar komponen serta merepresentasikan sistem tersebut dalam bentuk yang mempunyai tingkat abstraksi yang lebih tinggi [5]. *Reverse engineering* merupakan teknik merubah *source code* yang telah jadi menjadi model desain [8], *Reverse engineering* biasanya digunakan untuk mengetahui struktur dari program untuk direkayasa ulang dengan demikian akan dapat mendeteksi kesalahan atau masalah pada desain. *Sequence diagram* digunakan untuk menggambarkan interaksi antara *class* atau *objek class* dan juga bisa digunakan dalam aktifitas pengujian perangkat lunak[9].

Dengan membuat pola model *sequence diagram*, maka manfaat yang bisa didapatkan selain konsistensi antara desain dan implementasi terjaga adalah meminimalkan kesalahan dalam mendesain *sequence diagram*, terutama jika orang yang merancangnya belum pernah mengimplementasikan rancangan UML ke *source code*, Selain itu memudahkan penyusunan *sequence diagram*, karena bisa menggunakan *template* yang tersedia pada model *sequence diagram*. Untuk langkah ke depannya memungkinkan membuat aplikasi *smartphone* berdasarkan desain *sequence diagram*, tanpa harus melakukan pemrograman secara manual.

II. TINJAUAN PUSTAKA

A. *UML Sequence Diagram*

Sequence diagram memiliki fokus pada perilaku didalam sistem, mengilustrasikan bagaimana objek berinteraksi dengan objek lainnya. Didalam *sequence diagram* terdapat objek dan pesan yang dikirim antar objek. Biasanya *sequence diagram* digunakan untuk menggambarkan interaksi objek yang terjadi dalam suatu *use case*. Untuk satu *use case* hanya diperlukan satu *sequence diagram*, jika terdapat beberapa skenario dalam *use case* maka bisa ilustrasikan sebagai fragmen dalam sequence diagram [10].

Terdapat tiga jenis *stereotype* yang ada dalam *sequence diagram* menurut jacobson, yaitu <<boundary>>, <<entity >>, dan <<control>>. Elemen *boundary* adalah objek yang berinteraksi dengan aktor dari sistem, bisa berupa tampilan antarmuka pada monitor maupun kakas perangkat keras. Elemen *entity* merupakan objek yang merepresentasikan data dari sistem, sedangkan elemen *control* adalah objek yang bertugas mengelola alur logika dari elemen *entity* dengan elemen *control* serta mengelola interaksi antar elemen tersebut.

Dalam melakukan pengujian perangkat lunak, sequence diagram juga memiliki peran penting yaitu sebagai dasar membuat test case [11]. setiap pesan yang dikirim antar objek dalam *sequence diagram* merupakan interaksi antar objek yang bisa dievaluasi, pengujian dibuat berdasarkan pada pesan setiap objek yang bisa berupa *method* dalam *class* yang bisa dievaluasi dalam bentuk *test case*.

B. *Reverse Engineering* pada *UML Sequence Diagram*

Reverse engineering adalah proses menganalisa sistem dengan maksud mengidentifikasi komponen-komponen sistem dan keterkaitan/interaksi antar komponen serta merepresentasikan sistem tersebut dalam bentuk yang mempunyai tingkat abstraksi yang lebih tinggi [5]. *reverse engineering* merupakan praktek untuk menganalisa perangkat lunak secara sebagian maupun keseluruhan untuk mengektraksi informasi desain maupun implementasi.

Kegunaan dari *reverse engineering* adalah untuk melakukan analisis program, mendapatkan kembali arsitektur dan desain model UML dari aplikasi, serta visualisasi. Selain itu *reverse engineering* juga bisa dimanfaatan untuk menciptakan artefak seperti model desain untuk mendukung pembangunan perangkat lunak [12].

Reverse engineering pada perangkat lunak bisa dikelompokan menjadi dua yaitu statis dan dinamis, statis jika model diektraksi dari *source code* dan dinamis jika model didapatkan dari eksekusi program[13]. *Reverse engineering* secara statis cocok digunakan dalam pengembangan perangkat lunak yang berfungsi untuk memastikan kebenaran implementasi berdasarkan desain, sedangkan *reverse engineering* secara dinamis diekstraksi tanpa perlu mengetahui *source code* dan dapat mengambil informasi tentang pengelolaan *memory*, *code coverage*, dan *concurrency* pada program[14]. Melakukan *reverse engineering* pada *sequence diagram* bisa digunakan untuk memahami alur program dan digunakan untuk proses perawatan perangkat lunak[15]. *Reverse engineering* pada *sequence diagram* juga dilakukan pada aplikasi web untuk mengamati prilaku dinamis dari interaksi objek [16].

C. Konsistensi desain untuk sequence diagram

Konsistensi desain dan implementasi merupakan permasalahan yang sering muncul dalam pembangunan perangkat lunak [4][6-7], salah satu solusi untuk mengatasi ini adalah dengan menerapkan *reverse engineering* untuk menciptakan model yang dijadikan dasar untuk pembangunan perangkat lunak [12]. Luaran penelitian ini adalah model *sequence diagram* yang didapat dari proses *reverse engineering* sehingga bisa digunakan sebagai *template sequence diagram* untuk pembangunan perangkat lunak. Dengan model tersebut diharapkan konsistensi antara desain dan implementasi bisa dijaga, untuk itu perlu kriteria untuk mengukur konsistensi antara model *sequence diagram* dan implementasi *source code*. Tabel I merupakan kriteria mengukur konsistensi desain *sequence diagram* yang dikumpulkan dari [4][6-7].

TABEL I
KRITERIA KONSISTENSI DESAIN UML SEQUENCE DIAGRAM

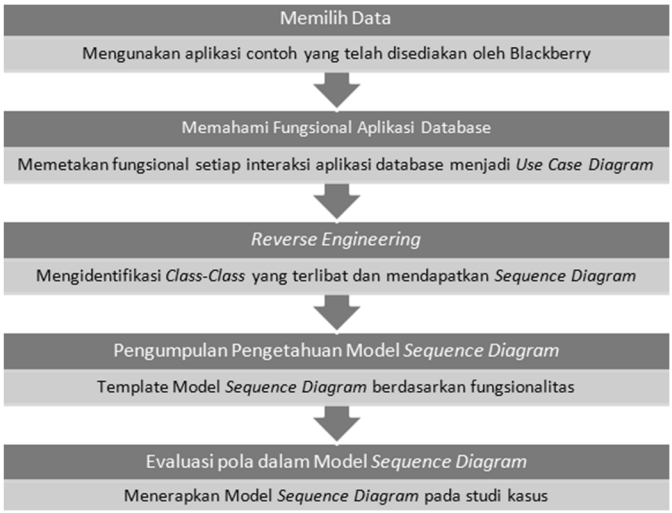
No	Kriteria	Sumber
1	Objek dalam sequence diagram sesuai dengan class diagram	[4]
2	Setiap objek harus paling tidak memiliki satu interaksi dengan objek lainya	[6]
3	Setiap pesan harus memiliki parameter yang sesuai	[6]
4	Nama dari pesan harus sama dengan operasi yang ada didalam class penerima	[7]
5	Arah pemangilan pesan harus sesuai dengan asosiasinya	[7]
6	Urutan pesan dari objek harus berdasarkan pada *events*	[7]

Pengukuran konsistensi dalam penelitian ini menggunakan enam kriteria, yaitu : Objek dalam *sequence diagram* sesuai dengan objek yang berada dalam *class diagram*, Setiap objek harus paling tidak memiliki satu interaksi dengan objek lainnya, setiap pesan harus memiliki parameter yang sesuai, nama dari pesan harus sama dengan operasi yang ada didalam *class* penerima pesan, arah pemangillan pesan harus sesuai dengan asosiasinya, dan urutan pesan dari objek harus berdasarkan pada *events* yang artinya semua operasi yang ada dipicu oleh aktor dalam pemanggilannya. Untuk mengukur tingkat konsistensi antara desain dan implementasi digunakan persamaan (1), berdasarkan kriteria dari tabel I diidentifikasikan kemungkinan ketidakcocokan yang mungkin terjadi dari desain *sequence diagram* dan ketidakcocokan yang benar-benar terjadi setelah implementasi desain *sequence diagram*.

$$Konsistensi = \left(1 - \frac{jumlah\ ketidakcocokan\ yang\ ditemukan}{jumlah\ ketidakcocokan\ yang\ mungkin\ ada}\right) x\ 100\% \quad (1)$$

III. METODE PENELITIAN

Seperti yang ditunjukan pada Gambar 1, tahapan yang dilakukan dalam melakukan penelitian ini adalah memilih *dataset*, memahami pola interaksi aplikasi basis data, kemudian melakukan *reverse engineering* untuk mendapatkan pola *sequence diagram*, lalu melakukan evaluasi dengan menerapkan pola *sequence diagram* tersebut pada studi kasus dan menguji konsistensi implementasi dari desain.



Gambar 1. Tahapan Penelitian

A. *Memilih Data*

Data yang digunakan adalah contoh source code yang tersedia di *repository online* blackberry (https://developer.blackberry.com/native/sampelapps/). Aplikasi yang dipilih adalah aplikasi yang berhubungan dengan pengelolaan data, hasil pemilihan tersebut bisa dilihat pada Tabel II.

TABEL II
DATASET APLIKASI

Aplikasi	Deskripsi	url
Data Creation	Basis data SQL	https://github.com/blackberry/Cascades-Sampels/tree/master/databasecreation
Database CRUD	Mengelola record dengan SQL	https://github.com/blackberry/Cascades-Sampels/tree/master/databasecrud
JSON Read and Write	Bekerja dengan data JSON	https://github.com/blackberry/Cascades-Sampels/tree/master/jsonreadwrite
Persistent Object	Menjadikan data persistent	https://github.com/blackberry/Cascades-Sampels/tree/master/persistentobjects
SOAP XML	Menggunakan SOAP webservice	https://github.com/blackberry/Cascades-Sampels/tree/master/soapxml

B. *Memahami Fungsional Aplikasi Basis Data*

Tahapan selanjutnya adalah memahami pola interaksi aplikasi yang ada dalam *dataset*. Aplikasi yang ada dalam *dataset* dijalankan untuk mengidentifikasi fungsionalitasnya. Fungsionalitas aplikasi basis data digambarkan dengan *use case diagram*.

C. *Reverse Engineering*

Setelah mendapatkan *use case diagram*, langkah selanjutnya adalah mengidentifikasi *class-class* yang terlibat dalam setiap *use case*, membuat *sequence diagram* sesuai *source code* yang tersedia.*Sequence diagram* dibuat berdasarkan objek *class* dan interaksi antar objek *class*.

D. *Pengumpulan Pengetahuan Model Sequence Diagram*

Pengumpulan pengetahuan dilakukan dengan cara mengamati pola-pola yang terbentuk berdasarkan karakteristik pola interaksi aplikasi basis data dengan *sequence diagram*. Misalnya berapa objek *class* yang terbentuk dalam *sequence diagram*, interaksi-interaksi apa saja yang dibutuhkan antar objek *class* dan lain-lain.

Hasil Pengumpulan pengetahuan adalah model interaksi *sequence diagram*. Model ini berupa *template* untuk membuat *sequence diagram*, sehingga *system analyst* memiliki panduan dan lebih mudah dalam membuat *sequence diagram* yang bisa di implementasikan secara konsisten ke dalam *source code*.

E. *Evaluasi pola dalam model sequence diagram*

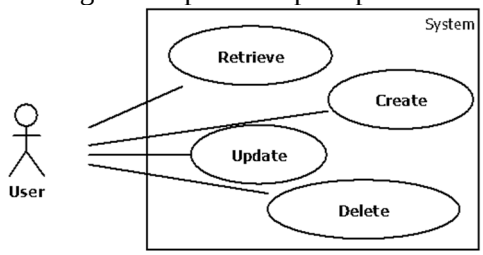
Evaluasi dilakukan dengan mengaplikasikan pola yang didapat pada suatu studi kasus untuk membuat aplikasi *smartphone* yang menggunakan basis data. Aplikasi yang dibuat sebagai studi kasus adalah sistem untuk menghitung efisiensi bahan bakar minyak (BBM) untuk kendaraan bermotor, aplikasi ini akan menyimpan data tentang *history* pengisian BBM yaitu berupa tanggal, jumlah liter BBM dan kilometer yang ditempuh oleh kendaraan. Dari penerapan aplikasi tersebut kemudian diukur konsistensi penerapan model *sequence diagram* pada studi kasus tersebut.

IV. HASIL

Sesuai dengan metode penelitian, hasil yang didapatkan untuk setiap tahapan adalah sebagai berikut :

A. *Fungsionalitas Aplikasi Basis Data*

Use case diagram didapatkan dari menganalisa interaksi yang disediakan pada antarmuka aplikasi. Gambar 2 adalah contoh salah satu *use case diagram* yang dihasilkan dari aplikasi "Database CRUD". Berdasarkan analisa interaksi didapatkan hasil pemetaan fungsional aplikasi seperti pada tabel 3.



Gambar 2. Use Case Diagram aplikasi "Database CRUD"

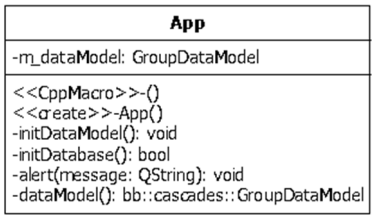
B. *Reverse Engineering*

Untuk setiap fitur yang teridentifikasi dari *use case* (tabel II) kemudian dibuat *sequence diagram* dengan cara mengidentifikasikan *class-class* menggunakan teknik *reverse engineering*. Alat-alat yang digunakan untuk melakukan *reverse engineering* adalah *Momentics IDE for Blackberry 10 Native SDK version 2.1* sebagai lingkungan pengembangan disertai *Blackberry 10 Device Simulator* menggunakan *VMWare Player*, serta *StarUML versi 5.0.2 .1570* untuk menggambarkan model *use case diagram, class diagram*, dan *sequence diagram*.

TABEL III
FUNGSIONAL APLIKASI

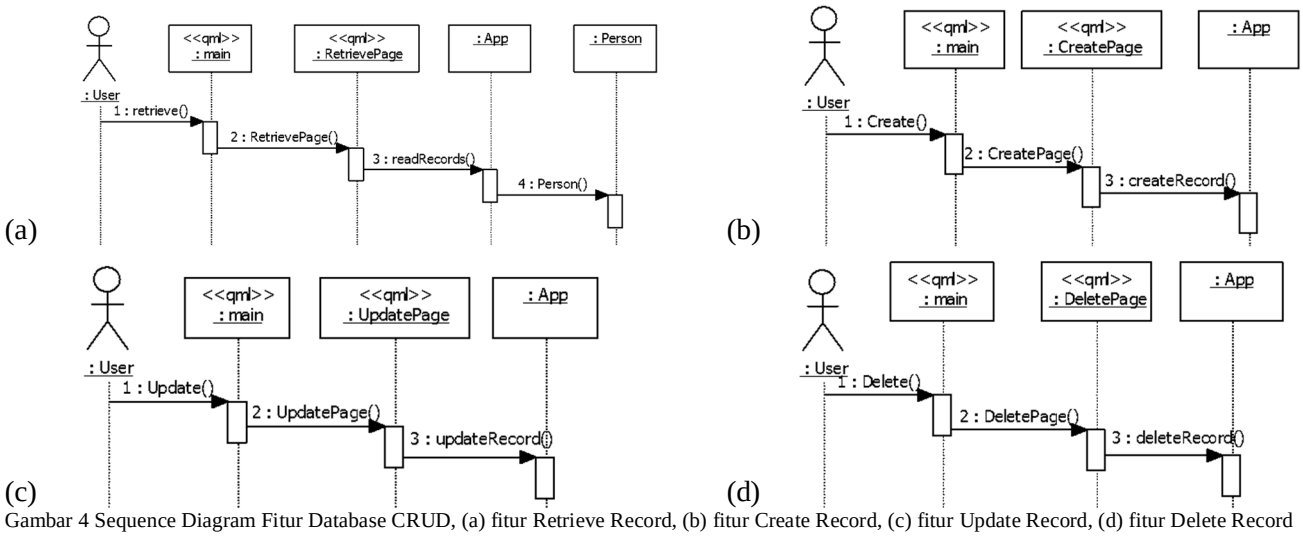
Aplikasi	Fitur	Deskripsi
Data Creation	*Create/Register Database*	Membuat basis data baru
	Create Table	Membuat tabel baru
	Create Table Asynchronously	Membuat tabel asynchronous baru
	Drop Table	Menghapus tabel baru
	Create Record	Membuat record baru
Database CRUD	*Retrieve*	Menampilkan record dari suatu tabel
	Create	Membuat record baru
	Update	Memperbarui record baru
	Delete	Menghapus record
JSON Read and Write	*Load JSON*	Membaca data JSON
	Convert JSON to Qt	Mengkonversi JSON ke format Qt
	Convert Qt to JSON	Mengkonversi Qt ke format JSON
	Write to JSON	Menulis data ke dalam JSON
Persistent Object	*Retrieve*	Menampilkan record dari suatu tabel
	Create	Membuat record baru
	Update	Memperbarui record baru
	Delete	Menghapus record
SOAP XML	*Request Weather Information*	Membaca informasi Web Service

Class-class yang terlibat diidentifikasikan dengan melakukan *reverse engineering* secara otomatis menggunakan kakas StarUML, hasil yang didapatkan adalah *class-class* yang dipakai dalam aplikasi beserta *attribut* dan *method* yang ada, gambar 3 adalah contoh hasil *reverse engineering* dari aplikasi "database CRUD".



Gambar 3 Salah satu *Class* yang didapatkan melalui proses *reverse engineering*

Setiap fitur yang berhasil diidentifikasikan pada tabel 3 merupakan *event* yang dipicu oleh pengguna aplikasi. Setiap *event* tersebut ditelusuri pemanggilan operasi(*method*) penggunaan *class* untuk digambarkan sebagai *sequence diagram*. Gambar 4 merupakan contoh *sequence diagram* yang dihasilkan dari tahapan ini.



Gambar 4 Sequence Diagram Fitur Database CRUD, (a) fitur Retrieve Record, (b) fitur Create Record, (c) fitur Update Record, (d) fitur Delete Record

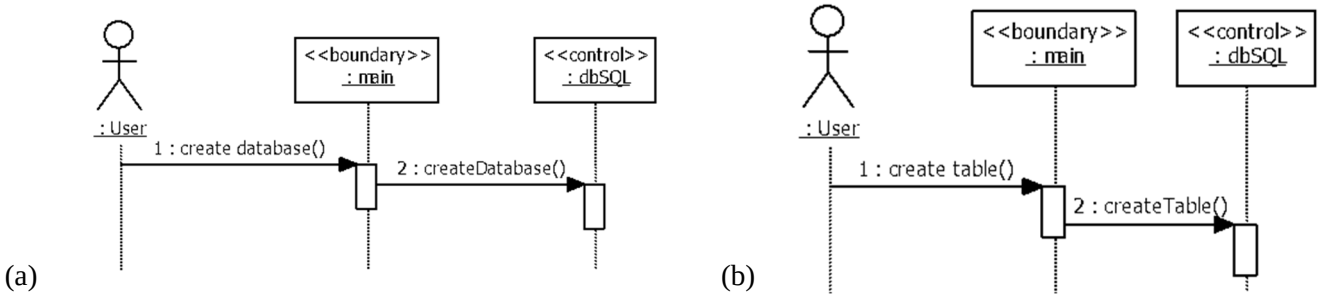
C. Model Sequence Diagram

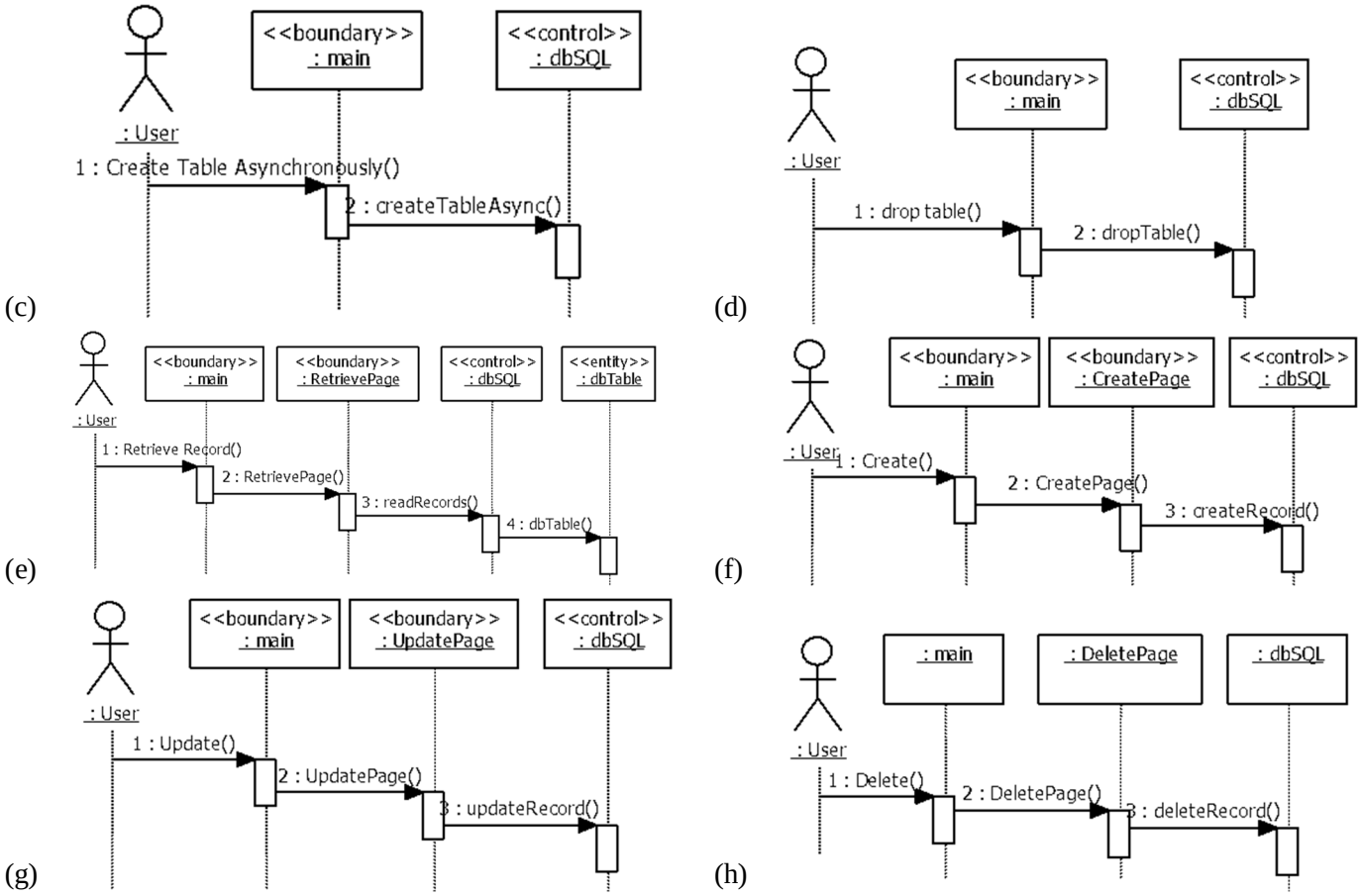
Fungsionalitas aplikasi yang telah diidentifikasikan pada tabel III kemudian dikelompokan sesuai dengan kesamaan fungsi, didapatkan tiga kelompok fitur yaitu : *SQL*, *Persistent Object*, dan *JSON & SOAP XML*. Kelompok *SQL* merupakan fitur yang berkaitan dengan pengelolaan DBMS seperti membuat tabel, menghapus tabel, menambah *record*, merubah *record*, atau menghapus *record* dalam tabel. Kelompok *Persistent Object* adalah fitur-fitur yang berkaitan dengan penyimpanan data pada file didalam *smartphone*. *JSON & SOAP XML* merupakan kelompok fitur yang pengelolaan basis data yang berinteraksi melalui internet. Pengelompokan fitur tersebut bisa dilihat pada tabel IV.

TABEL IV
POLA FITUR PENGELOLAAN BASIS DATA

Kelompok	Fitur	Deskripsi	Aplikasi Sumber
SQL	*Create/Register Database*	Membuat basis data baru	*Data Creation*
	Create Table	Membuat tabel baru	*Data Creation*
	Create Table Asynchronously	Membuat tabel asynchronous baru	*Data Creation*
	Drop Table	Menghapus tabel	*Data Creation*
	Retrieve Record	Menampilkan record dari suatu tabel	*Database CRUD*
	Create Record	Membuat record baru	*Data Creation Database CRUD*
	Update Record	Memperbarui record baru	*Database CRUD*
	Delete Record	Menghapus record	*Database CRUD*
Persistent Object	*Retrieve Record*	Menampilkan record dari suatu file	*Persistent Object*
	Create Record	Membuat record baru pada file	*Persistent Object*
	Update Record	Memperbarui record baru pada file	*Persistent Object*
	Delete Record	Menghapus record dari file	*Persistent Object*
JSON & SOAP XML	*Load JSON*	Membaca data JSON	*JSON Read and Write*
	Write to JSON	Menulis data ke dalam JSON	*JSON Read and Write*
	Request Web Service	Membaca informasi Web Service	*SOAP XML*

Gambar 5 merupakan hasil analisa pola *sequence diagram* yang dibuat berdasarkan fitur untuk kelompok *SQL*. Gambar 5a sampai 5h secara berurutan merupakan model *sequence diagram* untuk fitur membuat basis data baru, fitur membuat tabel baru, fitur membuat tabel *asynchronous* baru, menghapus tabel, menampilkan record dari suatu tabel, membuat record baru, memperbarui record baru, dan menghapus record. Model sequence diagram ini bisa menjadi panduan bagi *system analyst* untuk mendesain *sequence diagram* untuk aplikasi yang berkaitan dengan basis data pada *smartphone.*





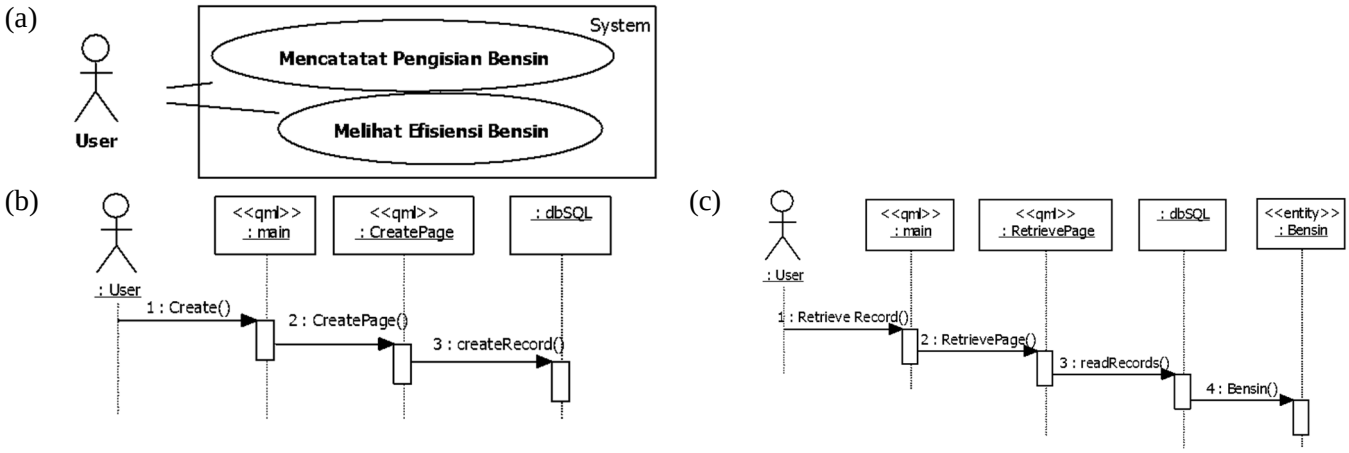
Gambar 5. Model *sequence diagram* (a) fitur membuat basis data baru. (b) fitur membuat tabel baru. (c) fitur membuat tabel asynchronous baru. (d) menghapus tabel. (e) menampilkan record dari suatu tabel. (f) membuat record baru. (g) memperbarui record baru. (h) menghapus record.

D. Evaluasi pola dalam model sequence diagram

1) Pembangunan Aplikasi untuk Studi Kasus

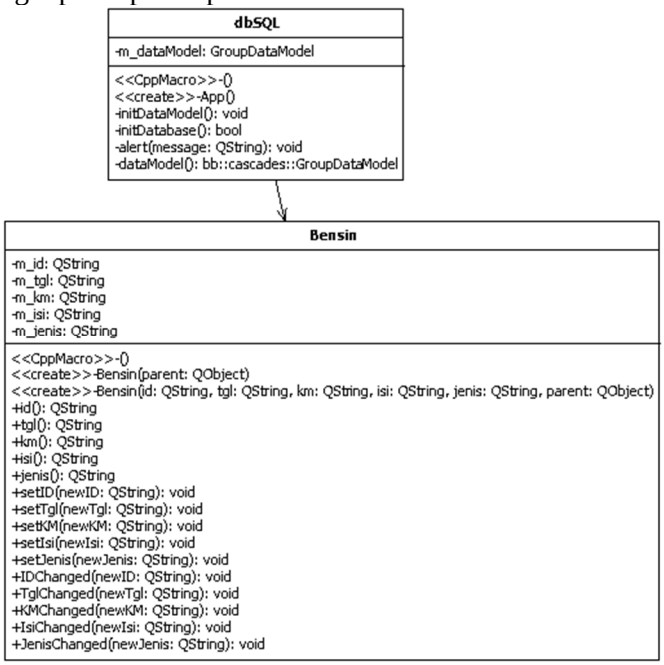
Untuk melakukan evaluasi terhadap hasil model *sequence diagram*, dibuat aplikasi *smartphone* Blackberry 10 yang menggunakan basis data dengan mengaplikasikan model *sequence diagram* yang telah dibuat sebagai studi kasus. Aplikasi yang dibuat adalah "EfisiensiBBM", yang memiliki kegunaan untuk menghitung efisiensi penggunaan bahan bakar minyak (BBM) pada kendaraan bermotor serta menyimpan data tentang *history* pengisian BBM berupa tanggal, jumlah liter BBM dan kilometer yang ditempuh oleh kendaraan.

Gambar 6 merupakan hasil dari analisa model yang dibuat sesuai dengan kebutuhan aplikasi berdasarkan Pola Fitur Pengelolaan basis data. Gambar 6a merupakan representasi *use case diagram* yang memiliki fitur "Mencatat Pengisisian Bensin" dan "Melihat Efisiensi Bensin". Gambar 6b dan 6c merupakan gambar *sequence diagram* dari dua fitur tersebut.



Gambar 6. Model yang digunakan untuk aplikasi "EfisiensiBBM" (a) model *use case diagram*, (b) model *sequence diagram* untuk pola fitur "Mencatat Pengisian Bensin". (c) model *sequence diagram* untuk pola fitur "Melihat Efisiensi Bensin".

Setelah membuat *use case diagram* dan *sequence diagram*, dapat ditentukan *class-class* yang terlibat dalam aplikasi ini, *class-class* tersebut bisa diturunkan dari Pola Fitur Pengelolaan basis data berdasarkan *class* yang dipakai pada *sequence diagram* maupun *class* yang dibuat sendiri oleh pembangun aplikasi. Gambar 7 merupakan *class diagram* yang dipakai pada aplikasi "EfisiensiBBM".



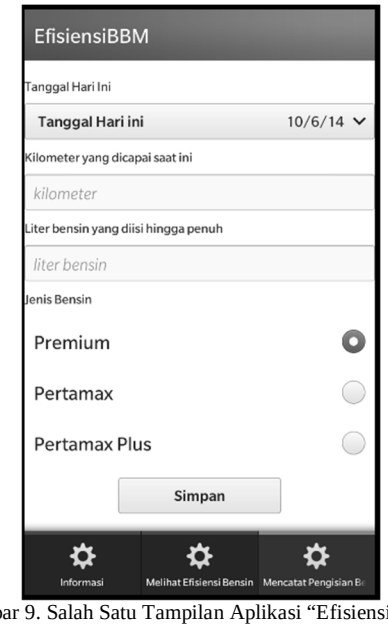
Gambar 7. *Class Diagram* "EfisiensiBBM"

Class-class tersebut diimplementasikan dalam kode, isi kode setiap *method* sesuai dengan isi *method* aplikasi contoh. Namun beberapa method masih perlu dilakukan penyesuaian terhadap code agar bisa sesuai dengan tujuan dari aplikasi ini. Gambar 8 merupakan implementasi code pada class "App" yang digunakan untuk aplikasi "EfisiensiBBM"

```
// CRUD Functions
bool App::createRecord(const QString &tgl,const QString &km,const QString &isi, const QString &jenis)

bool App::updateRecord(const QString &customerID, const QString &firstName, const QString &lastName)

bool App::deleteRecord(const QString &customerID)
//! [2]
void App::readRecords()
//! [3]
GroupDataModel* App::dataModel() const

// Alert Dialog Box Functions
void App::alert(const QString &message)

QString App::hitungEfisiensiBBM()
```

Gambar 8. implementasi Method dari *Class dbSQL* pada aplikasi "EfisiensiBBM"

Gambar 9 merupakan *screen shoot* dari tampilan aplikasi "EfisiensiBBM", kedua fitur dalam aplikasi ini dibuat menggunakan model UML *Sequence Diagram* berdasarkan pola fitur pengelolaan basis data sehingga lebih mudah karena me-*reuse* code hasil pengelompokan fitur berdasarkan *reverse engineering*. fitur "Mencatat Pengisisian Bensin" menggunakan pola fitur dan model *Sequence Diagram* "Create Record" sedangkan fitur "Melihat Efisiensi Bensin" menggunakan pola fitur dan model *Sequence Diagram* "Retrieve Record".



Gambar 9. Salah Satu Tampilan Aplikasi "EfisiensiBBM"

2) Pengujian Konsistensi Desain

Pengujian dilakukan dengan mengukur konsistensi desain dan implementasi dari aplikasi studi kasus yang mengaplikasikan model *sequence diagram* hasil penelitian ini. Konsistensi diukur berdasarkan kriteria konsistensi desain UML sequence diagram (tabel I) dengan menggunakan persamaan 1. Tabel V merupakan hasil penilaian konsistensi desain pada studi kasus yang menggunakan model *sequence diagram*, dalam tabel tersebut diindentifikasikan banyaknya resiko ketidakcocokan implementasi dan desain berdasarkan kriteria konsistensi serta menemukan jumlah ketidakcocokan yang benar terjadi. Berdasarkan jumlah ketidakcocokan tersebut bisa didapatkan nilai konsistensi desain dengan implementasi sesuai dengan kriteria yang dinilai.

3) Pelajaran dari Penerapan Model Sequence Diagram pada Studi Kasus

Berdasarkan studi kasus yang membutuhkan fitur yang telah disediakan oleh model *sequence diagram*, penerapan model tersebut pada aplikasi "EfisiensiBBM" dapat didapatkan beberapa pelajaran tentang dampak pengunaan *reverse engineering* terhadap konsistensi desain *sequence diagram*. Penerapan model *sequence diagram* berdasarkan kriteria konsistensi desain *sequence diagram* dapat menjaga konsistensi penerapan *class* dan *method* yang telah digambarkan pada *sequence diagram*. Berdasarkan studi kasus kriteria 1,2,4,5, dan 6 pada tabel V konsistensi bisa terjaga dengan menerapkan model *sequence diagram* dalam desain aplikasi tersebut, hal ini dimungkinkan karena pengetahuan operasi pengerjaan fitur tersebut telah didapatkan dan dipetakan dalam bentuk model *sequence diagram* sehingga pada penerapannya hanya perlu mengaplikasikan model tersebut pada desain *sequence diagram* aplikasi.

TABEL V
PERHITUNGAN KONSISTENSI DESAIN UML SEQUENCE DIAGRAM PADA STUDI KASUS

No	Kriteria	Ketidakcocokan yang mungkin ada *(a)*	Ketidakcocokan yang ditemukan *(b)*	Konsistensi *(1 – (b/a)) x 100%*
1	Objek dalam *sequence diagram* sesuai dengan *class diagram*	5	0	100%
2	Setiap objek harus paling tidak memiliki satu interaksi dengan objek lainya	5	0	100%
3	Setiap pesan harus memiliki parameter yang sesuai	7	1	85.72%
4	Nama dari pesan harus sama dengan operasi yang ada didalam *class* penerima	7	0	100%
5	Arah pemangilan pesan harus sesuai dengan asosiasinya	7	0	100%
6	Urutan pesan dari objek harus berdasarkan pada *events*	7	0	100%

Dalam studi kasus ini ditemukan bahwa pada kriteria ke 3 memiliki resiko ketidakcocokan antara desain dan implementasi, dalam studi kasus tersebut ketidakcocokan terjadi pada pemanggilan operasi / method pada class dengan *stereotype* <<entity>> karena pada class ini atribut, operasi, dan parameter dalam operasi tergantung dari desain yang dibutuhkan aplikasi. Dalam hal ini pola yang dibentuk dalam model *sequence diagram* hanya terbatas pada pola umum untuk mengerjakan suatu fungsionalitas dari aplikasi, jika ternyata model tersebut

membutuhkan pengaturan khusus agar sesuai dengan kebutuhan aplikasi maka resiko ketidakkonsistenan desain juga akan meningkat.

Selain itu model *sequence diagram* ini hanya terbatas pada fungsionalitas yang telah dimodelkan dari proses *reverse engineering*, Pola *sequence diagram* baru bisa diidentifikasikan dan dimodelkan jika telah ada aplikasi yang berjalan dan dilakukan analisa model *sequence diagram* dengan *reverse engineering*. Jika dalam pembangunan aplikasi menggunakan model *sequence diagram* namun terdapat beberapa fungsionalitas yang belum didukung oleh katalog model *reverse engineering* maka resiko ketidakkonsistenan desain dan implementasi akan meningkat.

V. KESIMPULAN

Penelitian ini mengusulkan untuk mengaplikasikan model *sequence diagram* untuk menjaga konsistensi antara desain *sequence diagram* dengan implementasi *source code*. Dalam penelitian ini model *sequence diagram* diperoleh dari proses *reverse engineering* pada *source code* aplikasi contoh yang ada dalam platform Blackberry 10. *Sequence diagram* hasil *reverse engineering* dikelompokkan sesuai dengan kesamaan fungsional fitur dan dimanfaatkan sebagai *template* pola *sequence diagram* yang digunakan sebagai alat bantu dalam membangun perangkat lunak. Berdasarkan hasil uji coba pada sebuah studi kasus, kriteria konsistensi desain *sequence diagram* seperti kesesuaian *class*, *method*, parameter, dan interaksi antara desain *sequence diagram* dan implementasi dapat dipertahankan dan memiliki nilai konsistensi yang tinggi.

Pengembangan penelitian selanjutnya yang bisa dilakukan adalah memadukan model *sequence diagram* ini dengan pola *source code* sesuai dengan operasi yang ada di class dalam *sequence diagram*, sehingga memungkinkan pembangunan perangkat lunak dengan otomatisasi implementasi berdasarkan *use case* dan *sequence diagram*.

DAFTAR PUSTAKA

[1] Kendall, K. E., Kendall, J. E.. “Systems Analysis And Design.” Edisi ke 7, Boston, Prentice Hall, 2008, hal 287.

[2] Dobing, Brian, and Jeffrey Parsons. "How UML is used." Communications of the ACM 49, nomor 5 halaman 109-113, 2006.

[3] Ferenc, R. “Modelling and Reverse Engineering C++ Source Code”, Disertasi Doctoral, Jurusan Informatics, University of Szeged, Szeged, Hungaria,2004.

[4] Li, Xiaoshan. "A Characterization of UML Diagrams and their Consistency" dipresentasikan dalam The 11th IEEE International Conference on Engineering of Complex Computer Systems , 2006 hal. 67-76.

[5] Chikofsky, E.J., Cross, J.H. “Reverse Engineering and Design Recovery : A Taxonomy” dalam IEEE Software, Vol. 7, issue 1 , hal 13 – 17, 1990.

[6] Thirugnanam, M., “An Efficient Design Tool to Detect Inconsistencies in UML Design Models.” International Journal of Computer Science and Business Informatics, Vol. 9 Ed. 1, hal. 36-44, 2014

[7] Egyed, A, “Instant consistency checking for the UML.” Dalam Proceedings of the 28th international conference on Software engineering ACM, hal. 381-390, Mei, 2006.

[8] Stringfellow, C., Amory, C. D., Potnuri, D., Andrews, A., & Georg, M.,"Comparison of software architecture reverse engineering methods." Information and Software Technology, Vol 48, No 7, hal. 484-497, 2006.

[9] Ziadi, T., Da Silva, M. A. A., Hillah, L. M., & Ziane, M. “A fully dynamic approach to the reverse engineering of UML sequence diagrams dipresentasikan dalam The 11th IEEE International Conference on Engineering of Complex Computer Systems, 2011, hal 107-116.

[10] Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., & Carsí, J. A. “Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments”. Information and Software Technology, Vol. 53 No.12, 2011, hal. 1391-1403.

[11] Swain, S. K., Mohapatra, D. P., & Mall, R. “Test case generation based on use case and sequence diagram”. International Journal of Software Engineering, Vol 3 No. 2,2010, hal. 21-52.

[12] CanforaHarman, G., & Di Penta, M. “New frontiers of reverse engineering”. In 2007 Future of Software Engineering, IEEE Computer Society, Mei, 2007, hal. 326-341

[13] Di Lucca, G. A., Di Penta, M., Antoniol, G., & Casazza, G. “An approach for reverse engineering of web-based applications”. Proceedings of Reverse Engineering Eighth Working Conference, IEEE, 2001, hal. 231-240.

[14] Systä, T., & Tamperensis, U. “Static and dynamic reverse engineering techniques for Java software systems”, Disertasi Doctoral, Jurusan Informatics, University of Tanpere, Tampere, Finlandia ,2000.

[15] Rountev, A., Volgin, O., & Reddoch, M. “Control flow analysis for reverse engineering of sequence diagrams” dalam Rapport Technique, Ohio State University, 2004.

[16] Alalfi, M. H., Cordy, J. R., & Dean, T. R. “Automated reverse engineering of UML sequence diagrams for dynamic web applications” dipresentasikan di Software Testing, Verification and Validation Workshops, IEEE, April, 2009 hal. 287-294.