

PENERAPAN *REVERSE ENGINEERING* DALAM PENENTUAN POLA INTERAKSI *SEQUENCE* *DIAGRAM* PADA SAMPEL APLIKASI ANDROID

Vierdy Sulfianto Rahmadani¹⁾, Indra Kharisma Raharjana²⁾, Taufik³⁾

Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Airlangga
Kampus C Mulyorejo, Surabaya

¹⁾vierdy-sr-10@fst.unair.ac.id

²⁾indra.kharisma@fst.unair.ac.id

³⁾taufik@fst.unair.ac.id

Abstrak—Tujuan penelitian ini adalah penerapan *reverse engineering* untuk penentuan pola interaksi *Sequence diagram* yang bisa digunakan oleh sistem analis sebagai pola *template* untuk mendesain UML *sequence diagram*. Aplikasi yang digunakan sebagai data dasar berasal dari aplikasi contoh milik Android, aplikasi inilah yang mengalami proses *reverse engineering* dan teridentifikasi polanya. Tahap pertama yang dilakukan dalam penentuan pola interaksi ini adalah pengumpulan aplikasi dataset. Tahapan selanjutnya adalah identifikasi fitur dan aktifitas aplikasi, melakukan *reverse engineering* sehingga didapatkan model *sequence diagram*, kemudian melakukan sistesis semua model tersebut menjadi pola interaksi *sequence diagram*. Langkah terakhir adalah menguji pola tersebut dengan menerapkannya dalam pembangunan aplikasi studi kasus. Berdasarkan hasil evaluasi, disimpulkan bahwa pola interaksi pada *sequence diagram* yang didapatkan dari penelitian ini dapat diterapkan pada perancangan perangkat lunak yang memiliki fitur-fitur yang sama dengan fitur-fitur yang terdapat pada penelitian ini.

Kata Kunci— *Reverse Engineering*, Pola Interaksi, *Sequence Diagram*, Android

Abstract—The purpose of this research is to apply the application of reverse engineering to determine interaction patterns of the Sequence diagram that can be used by system analysts as a template for designing UML sequence diagrams. Sample applications from android are used as dataset for reverse engineering and pattern identification. The first step is collecting application datasets. The next stage is identifying the features and applications activity, reverse engineering to obtain a sequence diagram model, and then synthesize all of the models into an interaction pattern of sequence diagram. The final step is to test the patterns by implementing it in an application development case stud. The evaluation results concludes that interaction patterns of sequence diagram designs obtained in reverse engineering steps is able to be implemented in software development that contained similar features with the obtained features in this research.

Keywords— *Reverse Engineering*, Interaction Pattern, *Sequence Diagram*, Android

I. PENDAHULUAN

Unified Modelling Language (UML) telah menjadi standar dalam pengembangan perangkat lunak secara baku. UML memiliki berbagai jenis diagram, yang dikategorikan menjadi dua kategori diagram, yaitu diagram yang berisi informasi struktural, dan diagram tentang informasi perilaku (Pressman, 2010). Permasalahan yang terjadi ketika menggunakan diagram UML adalah hal konsistensi antara diagram yang dibuat dan ketika implementasinya (Xiaoshan, 2006). ketidak konsistenan antara diagram desain dan implementasi akan berdampak pada terhambatnya proses pengembangan perangkat lunak maupun sulitnya perawatan perangkat lunak. Tidak konsistennya perangkat lunak bisa dibedakan menjadi dua yaitu tidak lengkapnya dokumen desain dan dokumen desain yang tidak sinkron dengan *source code* selama proses pembangunan perangkat lunak (Mens, Van Der Straeten, & Simmonds, 2003).

Untuk mengatasi permasalahan konsistensi antara diagram dan implementasinya, diusulkan untuk melakukan proses *reverse engineering* dari *working code* yang telah ada menjadi model diagram UML, dengan demikian model diagram yang dihasilkan merupakan representasi dari aplikasi yang telah jadi dan diharapkan memiliki konsistensi yang baik. *Reverse engineering* adalah sebuah cara mengubah *source code* menjadi model desain yang digunakan dalam rekayasa ulang sistem (Stringfellow, 2006). Tujuan utama dalam *reverse engineering* adalah untuk menghasilkan pandangan alternatif pada sistem, memulihkan kehilangan informasi, mendeteksi kesalahan sistem, mensintesis abstraksi sistem yang lebih tinggi, dan memfasilitasi *reengineering* (Tonella & Potrich, 2005).

Sebuah pola desain mempunyai tujuan tersendiri, yang juga mendeskripsikan *role*, *responsibility*, dan kolaborasi dari masing-masing kelas atau *instance* yang terlibat. Pola

dalam desain digunakan sebagai garis besar pengembangan aplikasi. Dengan ekstraksi pola desain dari *source code*, desain dan contoh dari sistem perangkat lunak didapatkan. (Shi & Ollson, 2006).

Dengan konsistensi dalam sebuah desain perancangan perangkat lunak, pengembangan perangkat lunak kedepannya dapat mengacu pada desain perangkat lunak. Pendekatan *Model-Driven Architecture* bisa dilakukan setelah penelitian ini. Pada penelitian ini, diterapkan proses *reverse engineering* pada sampel aplikasi Android. Tujuan penelitian ini untuk menemukan pola interaksi *sequence diagram*, agar konsistensi antara desain dan implementasi perangkat lunak dapat dicapai, melalui penerapan pola interaksi *sequence diagram* pada perancangan perangkat lunak.

II. TINJAUAN PUSTAKA

Software reengineering adalah pemeriksaan dan perubahan terhadap sebuah subyek sistem untuk menyusun kembali ke dalam sebuah bentuk yang baru dan implementasi yang sesuai dari bentuk yang baru tersebut. *Software reengineering* adalah proses rekayasa ulang yang mencakup 4 (empat) tujuan, yaitu *Understanding (predictive)*, *Repairing (corrective)*, *Improving (perfective)*, dan *Evolving (adaptive)*. *Reengineering* terdiri atas dua proses utama yaitu *reverse engineering* dan *forward engineering*.

Reverse Engineering merupakan proses yang tidak melibatkan perubahan pada sistem. sebuah sistem *software* dianalisis untuk mengekstrak informasi dari *software*, maka pilihan yang harus dilakukan adalah antara analisis *static* dan *dynamic* (Tonella & Potrich, 2005).

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu.

Dalam penelitian ini alat yang digunakan adalah *ModelGoon UML4Java*. *ModelGoon UML4Java* adalah sebuah *tools* atau *plugin* yang disematkan dalam IDE Eclipse untuk menggambarkan *class diagram* dan *sequence diagram*.

III. METODE PENELITIAN

Langkah-langkah penerapan *reverse engineering* untuk menentukan pola interaksi

Sequence diagram dalam perancangan sampel perangkat lunak sistem operasi Android adalah:

A. Pengumpulan Data Aplikasi

Proses ini mengumpulkan aplikasi berbasis android yang nantinya digunakan sebagai dasar penyusunan interaksi pola *sequence diagram*. Identifikasi dilakukan dengan uji *run* untuk mengetahui bahwa sampel tersebut adalah sampel yang termasuk *working code* (berhasil berjalan tanpa *error* pada *emulator* atau *device*).

B. Identifikasi Fitur

Setiap aplikasi mempunyai fungsi spesifik, yang disebut fitur. Fitur setiap aplikasi diidentifikasi untuk dibuat sebagai dasar pola interaksi dalam bentuk *sequence diagram*.

C. Identifikasi Sequence Diagram

Identifikasi *sequence diagram* sesuai fitur yang ada di dalam sampel dilakukan dengan bantuan *ModelGoonUML4Java plug-in* pada IDE Eclipse.

D. Menjabarkan Aktivitas Pada Setiap Fitur

Penjabaran aktifitas pada setiap *sequence diagram* dibutuhkan untuk memetakan langkah kegiatan pada setiap fitur berdasarkan uji coba menjalankan aplikasi dan *sequence diagram* yang telah diidentifikasi. Aktifitas ini nantinya digunakan sebagai dasar alur logika pembangunan pola interaksi dalam bentuk *sequence diagram*.

E. Analisis Pola Interaksi

Identifikasi pada pola interaksi *sequence diagram* dilakukan pada setiap fitur dan didasari oleh aktifitas yang sudah didapat sebelumnya. Analisis pola interaksi dilakukan untuk memetakan pola interaksi pada *sequence diagram*.

F. Evaluasi Pola Interaksi Sequence Diagram

Evaluasi pola dilakukan untuk menguji apakah pola interaksi pada fitur dari sampel perangkat lunak bisa diterapkan untuk merancang sebuah perangkat lunak, dengan mengambil pola interaksi yang sebelumnya dihasilkan. Hasil evaluasi berupa kesimpulan apakah pola interaksi tersebut dapat atau tidak dapat diterapkan untuk merancang perangkat lunak.

IV. PEMBAHASAN

A. Pengumpulan Data Aplikasi

Di bawah ini adalah sampel yang terkumpul:

1. AddVoicemailActivity
2. GameActivity & MainActivity (TicTacToe)
3. JetBoy
4. CubeWallpaperSettings
5. WalkieTalkieActivity

6. BroadcastReceiverNewSms
7. ContactManager

Dalam paper ini hanya akan membahas dua aplikasi contoh android saja yaitu *AddVoicemailActivity* dan *ContactManager*, hal ini dilakukan untuk mempermudah penyampaian hasil eksplorasi pola interaksi *sequence diagram* serta mempermudah proses evaluasi.

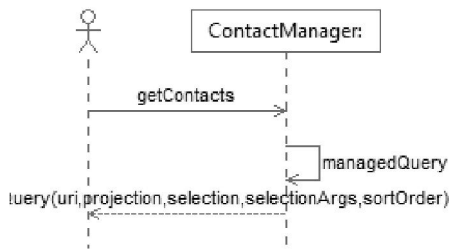
B. Identifikasi Fitur

Identifikasi fitur dilakukan sesuai fitur-fitur yang tersaji. Fitur-fitur yang terkumpul dapat dilihat pada Tabel 1.

C. Identifikasi Sequence Diagram

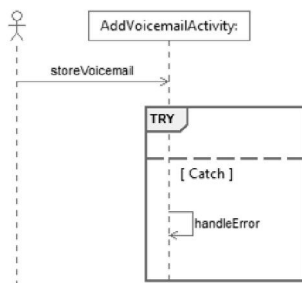
Beberapa contoh hasil pembangkitan *sequence diagram* menggunakan *ModelGoonUML4Java plug-in* adalah sebagai berikut :

1) *Contact Manager*: Fitur dari aplikasi ini adalah membaca kontak telepon serta menambah kontak baru. Identifikasi *sequence diagram* pada gambar 1 adalah fitur pengambilan kontak dari dalam *handphone*.



Gambar 1. Identifikasi *sequence diagram* untuk fitur pengambilan kontak dari *handphone*

2) *Voicemail*: Fitur dari aplikasi ini adalah merekam suara, menyimpan dan mengirim voicemail. Identifikasi *sequence diagram* pada gambar 2 berasal dari fitur *storevoicemail*.



Gambar 2. Identifikasi *sequence diagram* untuk fitur *store voicemail*.

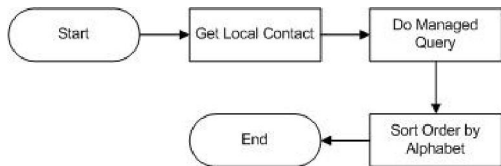
D. Menjabarkan Aktifitas Fitur Yang Terkumpul

Berikut merupakan contoh penggambaran algoritma pada fitur-fitur yang ada dalam aplikasi android, yaitu pengambilan kontak dari

telepon pada aplikasi *contact manager*, dan penyimpanan pesan suara pada aplikasi *voicemail*.

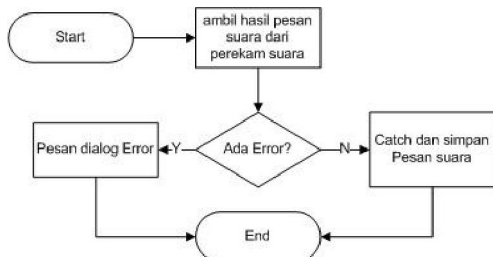
TABEL 1. HASIL IDENTIFIKASI FITUR

No.	Nama Perangkat Lunak	Keterangan	Fitur
1.	<i>Voicemail</i>	Sampel ini menunjukkan bagaimana merekam suara, menyimpan dan mengirim voicemail.	<ul style="list-style-type: none"> i. Merekam suara dengan <i>VoiceRecorder</i> ii. <i>Handling</i> hasil rekaman suara iii. <i>Storevoicemail</i>
2.	<i>TicTacToe</i>	Sampel yang tersedia ini berisi mesin utama <i>game</i> ini, berjalan ketika <i>user</i> memilih 2 pilihan dari <i>Playerfirst</i> atau <i>computerfirst</i>	<ul style="list-style-type: none"> i. Memproses masuk kepada <i>game</i> ketika user memilih <i>Playerfirst</i> atau <i>computerfirst</i> ii. <i>SensoringInput</i> ketika <i>user</i> memasukkan sesuai <i>tile</i> yang berlaku pada permainan. iii. Memproses <i>input</i> dari <i>user</i> dan menentukan pemenang.
		Sampel berisi aktivitas utama pada <i>game TicTacToe</i> , muncul pertama kali ketika program dijalankan.	<ul style="list-style-type: none"> i. <i>Startgame</i> memanggil <i>gameActivity</i>
3.	<i>JetBoy</i>	<i>Game</i>	<ul style="list-style-type: none"> i. Inisiasi program dengan memanggil dialog dan permainan ii. Sensor Sentuhan sesuai <i>keypressup</i>, <i>down</i> maupun <i>onclick</i>
4.	<i>Cube Wallpaper</i>	<i>DemoLiveWall paper</i> sederhana	<ul style="list-style-type: none"> i. Sensor Sentuhan ii. Generator grafis kubus/<i>decahedron</i>
5.	<i>Walkie Talkie</i>	<i>Demo</i> sederhana VoIP menggunakan akun SIP dan berkomunikasi seperti <i>WalkieTalkie</i>	<ul style="list-style-type: none"> i. <i>RegisterAccount</i> lewat SIP (<i>VoIP Provider</i>) ii. <i>Call</i> dan <i>Receivecall</i>
6.	<i>Broadcast Receiver New SMS</i>	Aplikasi sederhana untuk memberi notifikasi apabila ada SMS akan masuk	<ul style="list-style-type: none"> i. Mengambil informasi tentang SMS yang akan masuk ii. Menampilkan isi SMS
7.	<i>Contact Manager</i>	<i>Demo</i> pengelola kontak sederhana di telepon genggam	<ul style="list-style-type: none"> i. Membaca kontak dari telepon dan menampilkannya sesuai abjad ii. Menambah kontak baru



Gambar 3. Aktifitas pada pengambilan kontak di *local storage handphone*

1) *Sampel Contact Manager*: Pada gambar 3, ditunjukkan algoritma pengambilan kontak dari *local storage handphone*. Aktifitas dalam algoritma tersebut adalah melakukan pengambilan kontak lokal dan kemudian menampilkannya sesuai dengan proyeksi urutan.

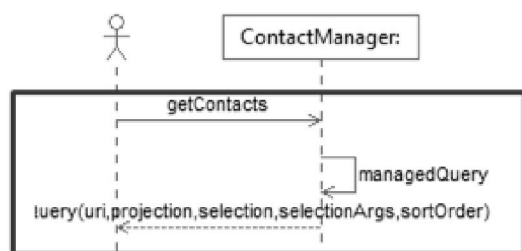


Gambar 4. Aktifitas pada penyimpanan pesan suara

2) *Sampel Voicemail*: Pada gambar 4, ditunjukkan algoritma *store voicemail*. Aktifitas dalam fitur ini adalah menyimpan suara dari perekam suara dan kemudian memeriksa apakah ada kesalahan atau tidak, jika ada menampilkan dialog *error*.

E. Analisa Pola Interaksi

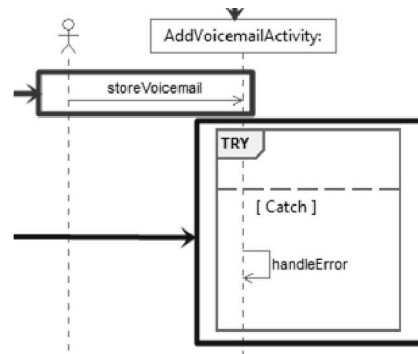
Aktifitas dan *sequence diagram* yang telah diidentifikasi pada tahap sebelumnya digunakan untuk melakukan analisa untuk penentuan pola interaksi. Hal tersebut dilakukan dengan meneliti pola-pola yang muncul pada *sequence diagram* pada setiap objek-objek yang tersedia dan dikaitkan dengan aktivitas pada fitur tersebut. Aktifitas dan *method* difilter dan dicari yang merupakan *method* yang utama dalam menjalankan suatu fitur, *method* tersebut kemudian digambarkan ulang dalam bentuk *sequence diagram* dan menjadi pola interaksi *sequence diagram* untuk menjalankan suatu fitur.



Gambar 5. Hasil identifikasi pola interaksi *sequence diagram* pengambilan kontak dari *handphone*

Contoh hasil identifikasi dan analisa pola interaksi *sequence diagram* ditampilkan pada gambar 5 dan gambar 6. Pada Tabel 2,

ditampilkan hasil pola interaksi yang berupa katalog. Katalog pola interaksi *sequence diagram* yang ditampilkan dalam paper ini merupakan hasil analisa dari 2 aplikasi contoh android yaitu *Voicemail&Contact Manager*.

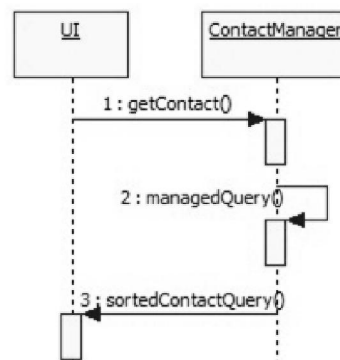


Gambar 6. Hasil identifikasi pola interaksi *sequence diagram* fitur *store voicemail*

F. Evaluasi Pola Interaksi Sequence Diagram

Evaluasi dilakukan dengan menggunakan pola interaksi *sequence diagram* yang diperoleh untuk diterapkan dalam studi kasus. Dalam evaluasi ini akan dibuat aplikasi yang memiliki kemampuan untuk mengirim pesan suara sesuai kontak yang dipilih. Aplikasi ini mengambil informasi kontak yang dipilih untuk dijadikan tujuan pengiriman pesan suara.

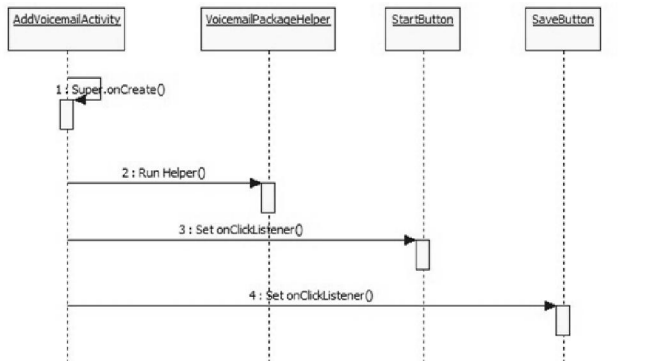
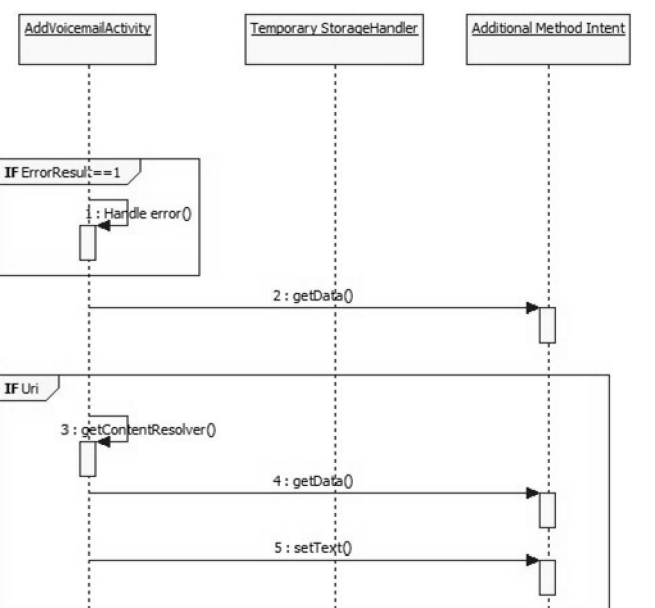
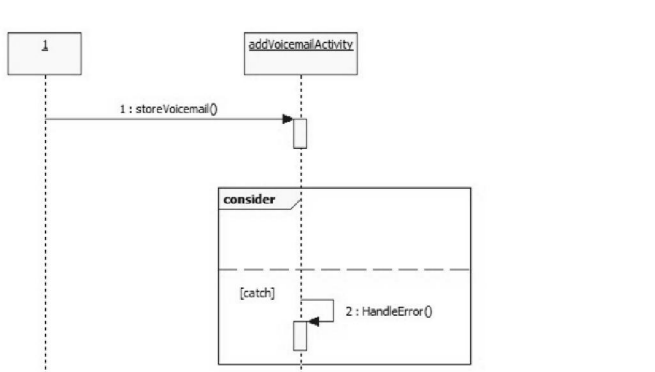

Aplikasi ini memanfaatkan pola interaksi *sequence diagram* yang telah diidentifikasi pada Tabel 2 untuk melakukan desain aplikasi menggunakan model *sequence diagram*, desain *sequence diagram* untuk aplikasi ini bisa dilihat pada gambar 7. Fitur yang digunakan dari pola interaksi *sequence diagram* adalah Membaca kontak dari telepon dan merekam suara.



Gambar 7. *Sequence Diagram* untuk aplikasi mengirim pesan suara sesuai kontak yang dipilih

Aktivitas dari aplikasi ini adalah menampilkan kontak telepon, kemudian pengguna bisa memilih salah satu kontak yang telah ditampilkan, dan bisa merekam *voicemail*. Pengguna kemudian bisa mengirim *voicemail* tersebut dengan tujuan kontak yang telah dipilih. Pada gambar 8 disajikan hasil *run* pada perangkat lunak untuk evaluasi.

TABEL 2.KATALOG POLA INTERAKSI

No.	Fitur	Keterangan	Pola Interaksi
1.	Merekam suara dengan Voice Recorder	Digunakan untuk merekam suara menggunakan bantuan voice recorder yang tersedia pada device.	 <pre> sequenceDiagram participant A as AddVoicemailActivity participant B as VoicemailPackageHelper participant C as StartButton participant D as SaveButton A->>A: 1: Super.onCreate() A->>B: 2: Run Helper() A->>C: 3: Set onClickListener() A->>D: 4: Set onClickListener() </pre>
2.	Handling hasil rekaman suara sebelum dikirim	Sebelum dikirim, fitur ini bertujuan untuk <i>handle</i> hasil rekaman sementara.	 <pre> sequenceDiagram participant A as AddVoicemailActivity participant B as Temporary StorageHandler participant C as Additional Method Intent A->>A: IF ErrorResult == 1 A->>A: 1: Handle error() A->>C: 2: getData() A->>A: IF Uri A->>B: 3: getContentResolver() A->>C: 4: getData() A->>C: 5: setText() </pre>
3.	Storevoice mail	Untuk mengambil hasil rekaman suara dan menyimpannya pada <i>storage</i> .	 <pre> sequenceDiagram participant A as 1 participant B as addVoicemailActivity A->>B: 1: storeVoicemail() B->>B: consider B->>B: [catch] B->>B: 2: HandleError() </pre>
4.	Membaca kontak dari telepon	Mengambil kontak yang ada dalam simcard dan device	 <pre> sequenceDiagram participant A as UI participant B as ContactManager A->>B: 1: getContact() B->>B: 2: managedQuery() B->>A: 3: sortedContactQuery() </pre>

TABEL 2. KATALOG POLA INTERAKSI (LANJUTAN)

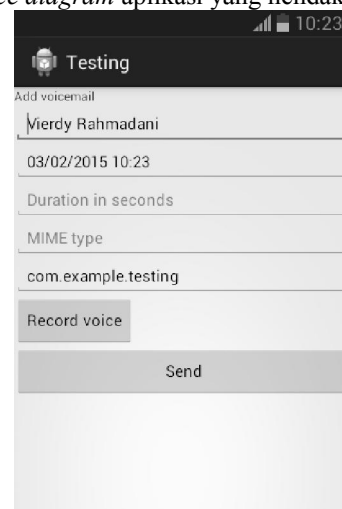
No.	Fitur	Keterangan	Pola Interaksi
5.	Menambah kontak baru	Menambah kontak, menyimpannya setelah user menginputkan pada form	<pre> sequenceDiagram participant UI participant ContactAdder participant Log UI->>ContactAdder: 1: btnSaveButtonClicked() activate ContactAdder ContactAdder->>Log: 2: send Log() deactivate ContactAdder ContactAdder->>UI: 3: call CreateContactEntry() deactivate ContactAdder UI->>ContactAdder: 4: finish() deactivate UI </pre>
6.	Menambah kontak baru	Menambah kontak diproses setelah tombol disentuh oleh user	<pre> sequenceDiagram participant CA as contact adder participant CTS as contact type spinner participant CR as ContactResolver participant CIF as Contact Info Fields participant B as builder participant T as Toast CA->>CTS: 1: getValue() activate CTS CTS->>B: 2: Build() activate B B->>CIF: 3: getValues() activate CIF CIF->>B: 4: build() deactivate CIF deactivate B CA->>CR: 5: getResolver() activate CR CR->>CA: 6: Apply() deactivate CR CA->>T: 7: getString() activate T T->>CA: 8: makeText() deactivate T CA->>CA: [Catch] </pre>

Dari hasil uji coba didapatkan bahwa pola interaksi *sequence diagram* yang telah dibuat bisa membantu dalam melakukan perancangan *sequence diagram* untuk aplikasi yang menggunakan fitur yang telah diidentifikasi pada pola interaksi *sequence diagram*. Analisis sistem bisa menggunakan *template* pola dalam merancang *sequence diagram* untuk kebutuhan perancangan sistem.

Solusi perancangan perangkat lunak dapat menggunakan pola interaksi yang telah didefinisikan, karena fitur yang dirancang memiliki kesamaan. Namun dengan penggunaan pola interaksi tersebut, dibutuhkan tambahan informasi *source code* perancangan aplikasi untuk setiap fiturnya. Beberapa *method* bisa menggunakan *source code* yang diambil dari aplikasi asal, terdapat kemungkinan dibutuhkan pula modifikasi pada *source code* tersebut. Namun fitur yang tidak tersedia pada pola interaksi *sequence diagram*, *source code* harus dilengkapi sendiri oleh programmer.

Keterbatasan dari penggunaan pola interaksi *sequence diagram* ini adalah pola fitur tersebut

harus telah memiliki aplikasi yang telah berjalan dan telah diidentifikasi polanya. Jika pola telah didefinisikan maka pola bisa diaplikasikan pada perancangan *sequence diagram* suatu aplikasi, namun jika belum maka analisis sistem tetap harus melakukan perancangan sendiri *sequence diagram* aplikasi yang hendak dibuat.



Gambar 8. Hasil run pada perangkat lunak untuk evaluasi

V. KESIMPULAN

Pola interaksi *sequence diagram* yang diidentifikasi dari tahapan penelitian ini, berdasarkan uji coba penerapan studi kasus dapat diterapkan sebagai *template* pola untuk mendesain *sequence diagram*. Sistem analisis maupun *programmer* bisa menggunakan *template* pola ini untuk memudahkan perancangan *sequence diagram*. Karena pola ini merupakan hasil *reverse engineering* dari fitur suatu aplikasi yang telah berjalan maka penerapannya memastikan akan konsistensi desain dan implementasi, namun berdasarkan hasil evaluasi studi kasus tidak menutup kemungkinan dibutuhkan modifikasi *source code* untuk memastikan kesesuaian pola fitur dengan tujuan aplikasi yang menggunakan *template* pola interaksi *sequence diagram* tersebut.

Integrasi pembangkitan *source code* berdasarkan pola interaksi *sequence diagram* bisa dilakukan untuk penelitian selanjutnya, karena *method* yang teridentifikasi ini merupakan *sourcecode* dari aplikasi yang telah berjalan. Penerapannya bisa dimanfaatkan untuk pengembangan metode *model-driven development*.

DAFTAR PUSTAKA

- Mens, T., Van Der Straeten, R., & Simmonds, J. (2003). Maintaining Consistency between UML Models with Description Logic Tools. *6th International Conference on the Unified Modeling Language – the Language and its applications (UML'2003)*, (hal. 71-77). San Francisco.
- Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach, 7th Edition*. New York: McGraw-Hill.
- Shi, N., & Ollson, R. A. (2006). Reverse Engineering of Design Patterns from Java Source Code. *21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006)*, 1-2.
- Stringfellow, C. (2006). Comparison of Software Architecture Reverse Engineering Methods. Dalam C. Stringfellow, *Information and Software Technology 48.6* (hal. 484-497).
- Tonella, P., & Potrich, A. (2005). *Monographs in Computer Science: Reverse Engineering of Object-oriented code*. Boston: Springer Science.
- Xiaoshan, L. (2006). A Characterization of UML Diagrams and Their Consistency. *Engineering of Complex Computer System, 11th IEEE International Conference*.

[This Page Intentionally Left Blank]