
TUGAS AKHIR - KI141502

# RANCANG BANGUN KAKAS BANTU DETEKSI KETIDAKSESUAIAN KODE SUMBER TERHADAP DIAGRAM URUTAN

ALBERT BUNGARAN MANIK
NRP. 5113 100 036

Dosen Pembimbing 1
Daniel Oranova Siahaan, S.Kom., M.Sc.,P.D.Eng.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

**TUGAS AKHIR – KI141502**

# RANCANG BANGUN KAKAS BANTU DETEKSI KETIDAKSESUAIAN KODE SUMBER TERHADAP DIAGRAM URUTAN

ALBERT BUNGARAN MANIK
NRP. 5113 100 036

Dosen Pembimbing 1
Daniel Oranova Siahaan, S.Kom., M.Sc.,P.D.Eng.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

***(Halaman ini sengaja dikosongkan)***


**FINAL PROJECT – KI141502**

# DESIGN AND IMPLEMENTATION OF CASE TOOL FOR INCOMPATIBILITY BETWEEN SOURCE CODE TO SEQUENCE DIAGRAM.

ALBERT BUNGARAN MANIK
NRP. 5113 100 036

Supervisor 1
Daniel Oranova Siahaan, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya 2018

*(Halaman ini sengaja dikosongkan)*

## LEMBAR PENGESAHAN

## RANCANG BANGUN APLIKASI DETEKSI KETIDAKSESUAIAN KODE SUMBER TERHADAP DIAGRAM URUTAN

**TUGAS AKHIR**

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

**Oleh:**
**ALBERT BUNGARAN MANIK**
**NRP. 5113100036**

Disetujui oleh Pembimbing Tugas Akhir:

1. Daniel Oranova Siahaan, S.Kom., M.Sc.,P.D.Eng.
   NIP. 197411232006041001 (Pembimbing 1)

**SURABAYA**
**JANUARI 2018**

*(Halaman ini sengaja dikosongkan)*

# RANCANG BANGUN APLIKASI DETEKSI KETIDAKSESUAIAN KODE SUMBER TERHADAP DIAGRAM URUTAN


**Nama** : **Albert Bungaran Manik**
**NRP** : **5113100036**
**Jurusan** : **Informatika**
**Fakultas Teknologi Informasi dan Komunikasi ITS**
**Dosen Pembimbing I** : **Daniel Oranova Siahaan, S.Kom, M.Sc., PDEng.**


## ABSTRAK


*Pada daur hidup perangkat lunak sendiri terdapat beberapa tahapan-tahapan yang harus dilalui yaitu, tahap pengumpulan kebutuhan,desain,implementasi,testing,perawatan. Fokus tugas akhir ini dititik beratkan pada tahap terakhir daur hidup perangkat lunak, yaitu tahap perawatan perangkat lunak. Dalam daur hidup tahap perawatan sangat rentan untuk mengubah kode sumber sesuai dengan kebutuhan atau keperluan fitur pada sebuah perangkat lunak yang mengakibatkan tidak sesuainya kode sumber terhadap desain awal. Tujuan tugas akhir ini yaitu untuk membuat sebuah kakas bantu untuk membantu analis sistem dalam mendeteksi ketidaksesuaian kode sumber tersebut terhadap desain, dalam hal ini desain yang menjadi pembanding adalah diagram urutan, hal ini dikarenakan diagram urutan menggambarkan alur sebuah fitur atau sistem.*

*Dalam tugas akhir ini kakas bantu akan dibentuk menggunakan bahasa pemrograman java. Diagram urutan dengan format xmi dan kode sumber dengan format xml merupakan input yang diperlukan untuk kakas bantu. Dalam tugas akhir ini pendeteksian ketidaksesuaian dilakukan menggunakan string matching dan word similarity. Kakas bantu ini nanti menghasilkan berupa sebuah alur dari diagram urutan yang ditunjukkan pada*

*tanda benar dan salah . Setiap alur digambarkan menjadi sebuah triplet. Triplet meliputi subyek,predikat,dan obyek.*

*Dalam program ini nanti akan diuji dengan data uji sebanyak 10 buah dataset yang berasal dari 5 buah kasus diagram urutan.Hasil program menunjukkan 95% kesepakatan terhadap analisis ahli rekayassa perangkat lunak*

***Kata kunci: Sequence Diagram,Kode Sumber , Deteksi diagram urutan ,ketidak sesuaian.***

**DESIGN AND IMPLEMENTATION CASE TOOL FOR INCOMPATIBILITY BETWEEN SOURCE CODE TO SEQUENCE DIAGRAM**


| | |
|---|---|
| **Name** | **: Albert Bungaran Manik** |
| **NRP** | **: 5113100036** |
| **Department** | **: Department of Informatics Faculty of Information Technology and Communication ITS** |
| **Supervisor I** | **: Daniel Oranova Siahaan, S.Kom, M.Sc., PDEng.** |


## ABSTRACT


*Software Development Life Cycle divides the development activities into planning stage, desing phase, implementation phase, and maintenance phase. In this final research, the research's focuson Software Development Life Cycle (SDLC) is on maintenance phase. In the Software Development Life Cycle , maintenance phase is very crucial. It is crucial because it most likely to change the code structure or modify the code based on the new requirement.*

*In this final research, case tool will be formed by using java programming language. Case tool's input must be in xmi for sequence diagram and xml for source code.In this final research, the detection of incompability use string matching and word similarity method. This case tool's output will produce a triplet which consist subyek, predikat, and obyek. The result will mark the each subyek, predikat, obyek with true and false sign.*

*The testing's result of this case tool show that 95% case tool's result accepted by software expert with 10 pieces dataset from 5 sequence diagram.*



***Keyword:. Sequence diagram, source code, incompatibility, incompatibility detection***

*(Halaman ini sengaja dikosongkan)*

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan YME karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

**“Rancang Bangun Aplikasi Kode Sumber terhadap Diagram Urutan”**

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Tuhan YME atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Papa, mama,dan adik penulis yang tiada henti-hentinya memberikan semangat, perhatian dan doa selama perkuliahan penulis di Jurusan Teknik Informatika ini.
3. Bapak Daniel Oranova Siahaan, S.Kom,M.Sc,PDEng selaku dosen pembimbing yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Saudara dari keluarga penulis yang selalu memberi semangat untuk menyelesaikan perkuliahan.
5. Teman-teman dari penulis yaitu Ari, andrew, andre, david, billy, jandre, romario, yosua, dan freddy yang sudah mau mensupport penulis baik dari moral dan pikiran.
6. Seluruh pihak yang tidak bisa saya sebutkan satu persatu yang telah memberikan dukungan selama saya menyelesaikan tugas akhir ini.

Mohon maaf apabila terdapat kekurangan dalam penulisan buku tugas akhir ini. Kritik dan saran sangat diharapkan untuk

perbaikan dan pembelajaran di kemudian hari. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Januari 2018

Penulis

# DAFTAR ISI

*(Halaman ini sengaja dikosongkan)*

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR KODE SUMBER

*(Halaman ini sengaja dikosongkan)*

# BAB I
# PENDAHULUAN

Pada bab ini akan dijelaskan hal-hal yang menjadi latar belakang,permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

## 1.1 Latar Belakang

Pada sebuah pengembangan perangkat lunak, terdapat daur hidup yang membagi tahapan aktifitas pengembangan perangkat lunak. Tahapan aktivitas tersebut meliputi tahap perancangan, tahap analisis, tahap desain, tahap pembuatan, tahap pengujian, dan tahap perawatan perangkat lunak. Fokus dari penelitian proposal ini terletak pada tahapan perawatan perangkat lunak. Tahap perawatan perangkat lunak merupakan tahap dimana perangkat lunak dimodifikasi sistem atau komponennya setelah penggunaan oleh konsumen untuk memperbaiki kerusakan,dan meningkatkan kinerja dari perangkat lunak tersebut. Tahap perawatan perangkat lunak merupakan tahap terakhir dari proses daur hidup perangkat lunak. Tahap ini merupakan tahap yang sangat rentan untuk memodifikasi kode sumber untuk bisa memenuhi kebutuhan pengguna. Tentu perubahan kode sumber kedepannya pasti mempengaruhi perangkat lunak dari desain awal perangkat lunak tersebut. Maka dari itu penelitian ini dilakukan untuk bisa membantu pihak pengembang perangkat lunak agar dapat mendeteksi ketidaksesuaian kode sumber mereka dengan desain yang mereka milikki. Pada kasus penelitian ini, desain yang digunakan sebagai pengukur ketidaksesuaian adalah diagram urutan. Diagram urutan dipilih menjadi pengukur karena diagram urutan menggambarkan bagaimana sebuah alur fitur atau sistem berinteraksi. Dari alur tersebut nantinya akan dideteksi ketidaksesuaian dari kode sumber.

Dari penelitian terkait yang dilakukan oleh rujukan [7] dilakukan sebuah pendeteksian untuk mendeteksi perubahan kode sumber yang telah dimodifikasi untuk mendeteksi *plagiarism* (plagiat) dengan menggunakan *similarity detection running karp rabin* yang menunjukkan hasil deteksi *plagiarism* dari perubahan kode sumber.

Dalam tugas akhir ini, tugas akhir akan melakukan penelitian yang merujuk dari penelitian terkait [7] dimana penelitian melakukan pendeteksian ketidaksesuaian diagram urutan. Tugas akhir ini nanti akan melakukan pendeteksian ketidaksesuaian kode sumber terhadap diagram urutan yang diharapkan bisa menghasilkan sebuah hasil yang menunjukkan ketidaksesuaian yang terdapat dalam kode sumber.

## 1.2 Rumusan masalah

Rumusan masalah yang diangkat dalam pembuatan perangkat lunak ini dipaparkan sebagai Berikut:

1. Apakah hasil dari kakas bantu ini berhasil mendeteksi ketidaksesuaian kode sumber terhadap diagram urutan ?
2. Bagaimana hasil data uji kakas bantu dan ahli ?
3. Bagaimanakah hasil respon dari uji non-fungsional ?
4. Bagaimanakah cara mengekstraksi diagram urutan dan kode sumber menjadi masukkan kakas bantu ?

## 1.3 Batasan masalah

1. Perkakas bantu di buat merupakan aplikasi berbasis dekstop
2. Pendeteksian dilakukan untuk bahasa berbasis pemrograman berbasis *object* yaitu *java*.
3. Tipo (salah ketik) pada nama fungsi dan metode mempengaruhi hasil perkakas bantu.

4. Kakas bantu hanya mengolah pesan berbentuk *message* pada UML
5. Inputan diagram urutan kakas bantu berbentuk *xmi*, menggunakan ekstraksi *plugin* dari Star-UML
6. Inputan kode sumber berbentuk xml, menggunakan ekstraksi *xmltranslatorapp*

## 1.4 Tujuan

Tujuan dari pembuatan perangkat ini adalah membuat alat kakas bantu untuk bisa mendeteksi ketidaksesuaian kode sumber dengan diagram urutan.

## 1.5 Metodologi

Ada beberapa tahapan dalam pengerjaan tugas akhir ini,yaitu sebagai berikut:

1. Penyusunan proposal tugas akhir
   Proposal tugas akhir berisi gagasan dari Tugas Akhir yang akan dikerjakan. Proposal Tugas Akhir berisi pendahuluan yang terdiri dari latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah,tujuan ,dan manfaat dari pembuatan tugas akhir. Selain itu, tinjauan pustaka yang digunakan juga dijabarkan sebagai refrensi pendukung pengerjaan Tugas Akhir. Subbab metedologi berisi penjelasan mengenai tahapan penyususunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.
2. Studi literatur
   Studi literatur yang dipelajari untuk mendukung pengerjaan Tugas akhir antara lain diagram urutan, *XML meta data interchange*, *starUML*, *xmltranslator*, *Saxparser*, *string matching*, dan *word similarity*.
3. Analisis dan perancangan sistem

Analisis dan perancangan sistem yang digunakan berdasarkan pengembangan perangkat lunak berbasis objek. Analisis dan perancangan diawali dengan analisis dan pendefenisian kebutuhan sistem terhadap masalah yang sedang dihadapi. Tahap perancangan dibagi menjadi beberapa tahap,yaitu:

1. Perancangan diagram UML
2. Perancangan antar muka perangkat lunak
3. Perancangan proses perangkat lunak

4. Implementasi sistem
   Tahap ini melakukan implementasi sistem perangkat lunak. Implementasi didasarkan pada perancangan sistem yang sudah dirancang sedemikian rupa sebelumnya. Perangkat lunak diimplementasikan dengan bahasa pemrograman *java*..

5. Pengujian dan evaluasi
   Pengujian dan evaluasi perangkat lunak yang telah dibuat dilakukan pengujian data uji (*testing data*) dengan menguji data diagram urutan yang ada terhadap kode sumber. Hasil testing akan berupa sesuai atau tidak nya diagram urutan terhadap kode sumber yang akan digambarkan melalui indikasi tanda pada setiap hasil testing. Setelah itu dlakukan pengujian dari ahli, dimana ahli akan melakukan pendeteksian secara manual. Nantinya hasil ahli akan dibandingkan dengan hasil kakas bantu.

## 1.6 Sistematika Penulisan

Penulisan buku tugas akhir ini dibagi kedalam 6 bab yang masing-masing menjelaskan bagian-bagian yang berbeda namun tetap memiliki korelasi satu dengan yang lain, yaitu:

1. Bab I, Pendahuluan, berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi serta sistematika penulisan buku.
2. Bab II, Dasar Teori, berisi penjelasan teori-teori yang digunakan sebagai dasar pengerjaan tugas akhir ini.

3. Bab III, Analisis dan Perancangan Sistem, berisi rancangan pembuatan sistem penyelesaian permasalahan dalam tugas akhir ini.
4. Bab IV, Implementasi, berisi lingkungan serta hasil penerapan rancangan sistem penyelesaian permasalahan dalam tugas akhir ini dalam bentuk sumber kode beserta penjelasannya.
5. Bab V, Pengujian dan Evaluasi, berisi lingkungan serta hasil dari rangkaian uji coba yang dilakukan untuk menguji kebenaran serta kinerja dari sistem.
6. Bab VI, Kesimpulan dan Saran, berisi kesimpulan pengerjaan tugas akhir ini dan saran untuk pengembangan kedepannya.
7. Daftar Pustaka,merupakan daftar referensi yang digunakan dalam pengembangan Tugas Akhir.
8. Lampiran,merupakan bab tambahan yang berisi daftar istilah yang penting dalam aplikasi ini.

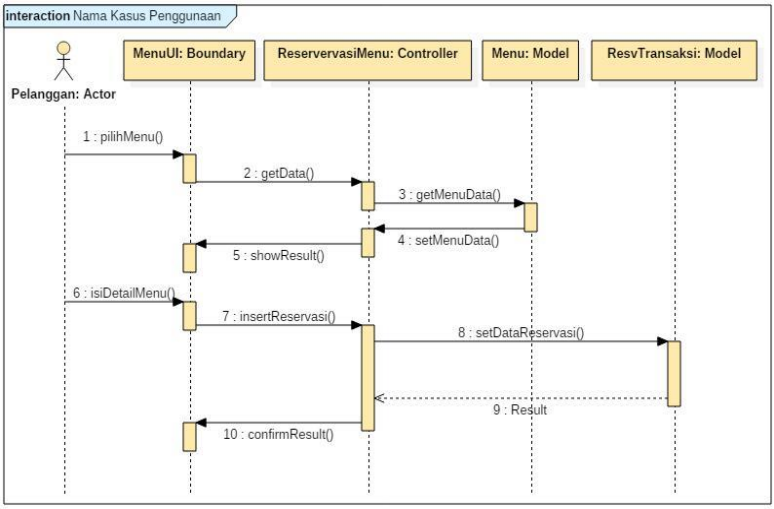
***(Halaman ini sengaja dikosongkan)***

# BAB II
# DASAR TEORI

Bab ini berisi dasar teori, publikasi ilmiah, dan kakas bantu atau pustaka perangkat lunak yang digunakan untuk menunjang pembuatan perangkat lunak.

## 2.1 Diagram Urutan

Diagram urutan merupakan sebuah diagram interaksi yang menunjukkan bagaimana sebuah *object* dapat berinteraksi dengan satu sama lain. Interaksi ini ditunjukkan dalam sebuah skenario dimana skenario ini menggambarkan interaksi object setiap waktu. Tujuan dari pembuatan diagram urutan adalah sebagai alat pelacak kesesuaian logika interaksi antara object–object di dalam sistem sesuai dengan urutan waktu interaksi tersebut terjadi. Salah satu contoh dari diagram urutan (*sequence diagram*) digambarkan pada Gambar 2.1




**Gambar 2.1 Contoh Diagram Urutan**

Komponen diagram urutan dijelaskan pada Tabel 2.1

**Tabel 2.1 Tabel komponen diagram urutan**

| Nama Komponen | Notasi | Keterangan |
|---|---|---|
| **Aktor** |  | Menggambarkan entitas eksternal |
| **Objek** |  | Menggambarkan sebuah objek dalam suatu sistem.bagian "*MenuUI"* merupakan nama objek |
| **Lifeline** |  | Menggambarkan daur hidup sebuah objek |
| **Activation Bar** |  | Menggambarkan durasi atau lamanya sebuah pesan |
| **Message** |  | Menggambarkan pesan(*message*) untuk komunikasi antar objek. |
| **Interaction** |  | Menggambarkan sebuah dasar dari diagram urutan untuk melektakkan komponennya |

Pada Gambar 2.1 Terdapat 2 buah kasus,yaitu:

1. Aktor memilih menu pada halaman(*boundary*) menu UI, setelah aktor memilih menu pada halaman *MenuUi*, *MenuUi* mengirim atau memilikki metode untuk *getData()* pada controller *reservasiMenu*. lalu *controller* akan me*request* melalui metode *getMenuData()* pada model menu yang nanti nya akan me*return setMenuData()* pada *controller* reservasi dan akan menampilkan *showResult()* pada *boundary MenuUi.*
2. Aktor mengisi detail menu melalui metode *isiDetailMenu()* melalui *boundary menuUi* yang nantinya di dimasukkan oleh metode *insertReservasi* melalui *controller reservasiMenu*. Saat controller menjalakan insert controller *reservasiMenu* akan menset data pada model *resvTransaksi* melalui method *setDataReservasi*, nantinya model akan me*return* hasil pada controller *reservationMenu* yang nantinya akan menampilkan *confirmResult* pada *boundary MenuUI*

## 2.2 Penelitian terkait

Penelitian terkait menjadi salah satu acuan penulis dalam melakukan penulisan tugas akhir. Dari penelitian terkait, tidak ditemukan penelitian dengan judul yang sama seperti judul tugas akhir. Namun penulis mengangkat beberapa penelitian sebagai refrensi dalam memperkaya bahan kajian sebagai refrensi tugas akhir. Berikut penelitian terkait dengan tugas akhir.

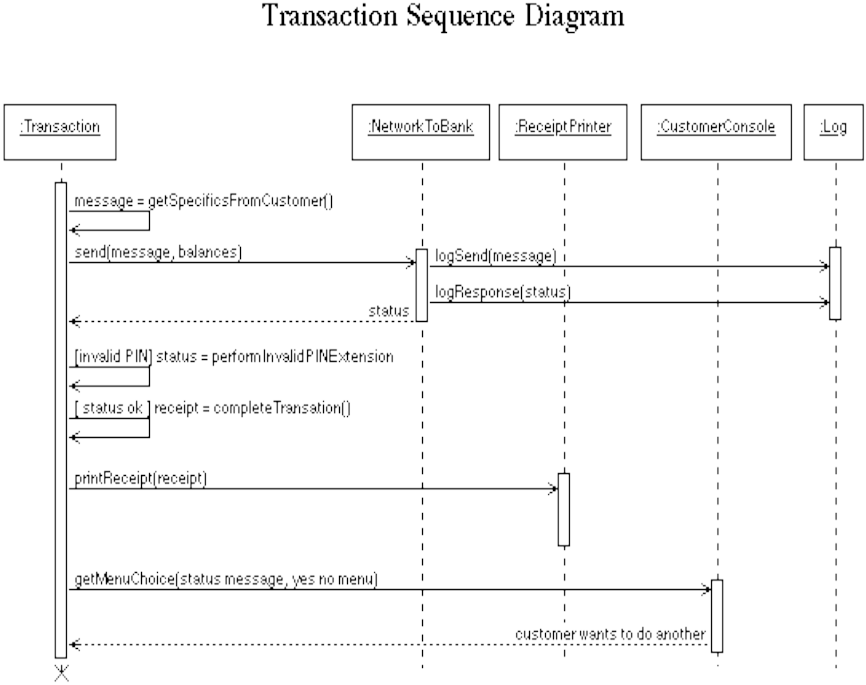
**Tabel 2.2 Penelitian terkait tugas akhir**

| Nama Peneliti | Judul Penelitian |
|---|---|
| Djuric Zoran , Gasevic Dragan | A Source Code Similarity System for Plagiarism Detection |

## 2.3 Ketidaksesuaian kode sumber

Ketidaksesuaian antara diagram urutan dan kode sumber bisa dilihat dari ada atau tidak nya nama metode dan kelas yang di cocokan berdasakan diagram urutan yang sudah dibuat. Dengan mengetahui terdapat kecocokan metode dan kelas antara kode sumber dan diagram urutan, kita dapat mengetahui ketidaksesuaian antara kode sumber yang dibentuk dengan perancangan sistem melalui diagram urutan. Contoh kasus ketidak sesuaian diagram urutan dan kode sumber adalah sebagai berikut.

Salah satu contoh kasus diagram urutan transaksi pada ATM (*Automatic Teller Machine*). Pada Gambar 2.2




**Gambar 2.2 Contoh transaksi ATM pada *sequence diagram***

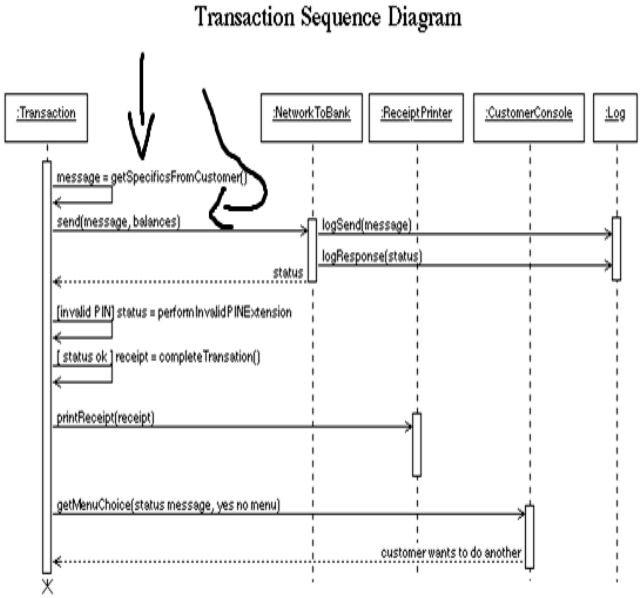
Dari Gambar 2.2 diagram urutan menunjukkan bagaimana setiap *object* dari *transaction.java* berinteraksi satu sama lain. Contoh kode sumber untuk fungsi transaction yang dimana sequence diagram tidak sesuai dengan kode sumber di gambarkan pada Gambar 2.3.

```
Public boolean Transaction() throws CardRetained
                {
                        .....

                         while (true){
                        switch(state)
                        {
                        .....
                        try
                                {

                         atm.getCustomerConsole().display("");
                                state = SENDING_TO_BANK_STATE;
                                }
                        catch(CustomerConsole.Cancelled e)
                        .....

               case SENDING_TO_BANK_STATE:


                    ......
                        else
                    {
                       state = ASKING_DO_ANOTHER_STATE;
                    }

                    break;
```

**Gambar 2.3 Contoh Kode Sumber (Source Code)**

Dari kode sumber pada Gambar 2.3 di atas di dapati kode sumber tidak sesuai dengan diagram urutan yang diberikan, karena kode sumber tidak memilikki sebuah metode *getSpecificsFromCustomer()* dan metode status = *atm.getNetworkToBank().sendMessage(message,balances)*.
Karena dalam Transaction tidak terdapat 2 metode tersebut maka diagram urutan dikategorikan tidak sesuai dengan kode sumber

yang ada. Berikut hasil setelah pendeteksian ketidak sesuaian kode sumber dapat diilustrasikan melalui alur diagram urutan seperti Gambar 2.4




**Gambar 2.4 Hasil setelah pendeteksian ketidaksesuaian diagram urutan**

## 2.4 StarUML

Star Uml adalah sebuah tools *Unified Modeling Language*(UML) yang di kembangkan oleh *MKLab*. *StarUML* sendiri merupakan sebuah applikasi berbasis dekstop yang digunakan untuk membuat diagram *Unified Modeling Language* (UML). Dalam kakas bantu ini nantinya diagram urutan akan dibentuk menjadi *xmi*.

## 2.5 Pemrograman Berbasis Object

Pemrograman berorientasi objek merupakan sebuah metode pemrograman yang berorientasi kepada objek. Tujuan dari pemrograman berorientasi objek diciptakan untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada.

## 2.6 Xmltranslator

*XMLtranslator* merupakan sebuah program berbasis web yang memetakan sebuah kode sumber berbasis object ke dalam sebuah AST (Abstract Syntax Tree). Hasil dari *xmltranslator* berupa file *xml*. Dalam aplikasi ini nantinya kode sumber akan dibentuk menjadi *xml*.

## 2.7 String Matching

```
public boolean    equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof String) {
        String anotherString = (String)anObject;
        int n = count;
        if (n == anotherString.count) {
            char v1[] = value;
            char v2[] = anotherString.value;
            int i = offset;
            int j = anotherString.offset;
            while (n-- != 0) {
                if (v1[i++] != v2[j++])
                    return false;
            }
            return true;
        }
    }
    return false;
}
```

**Gambar 2.5 Gambar potongan kode string matching metode**

Metode *string matching* digunakan untuk melakukan pencarian semua kemunculan string. Pada tugas akhir ini Algoritma string matching yang digunakan adalah fungsi *equals* pada pustaka yang telah disediakan oleh java. Gambar 2.5 menunjukkan potongan

metode dari *equals*. Metode ini digunakan untuk mendeteksi persamaan dari dua buah string yang ada, yang berguna untuk menemukan kesamaan antara dua buah string.

## 2.8 WordNet

*WordNet* adalah sebuah basis data leksikal dalam bahasa inggris. *WordNet* ini nantinya mendeskripsikan makna suatu kata dan mengelompokkan kata sesuai abjad. Kata yang berdekatan memilikki makna yang berhubungan. Dalam kasus tugas akhir ini nantinya setiap kata atau kalimat yang ditemukan akan dihitung kemiripannya. Perhitungan kemiripan tersebut dilakukan dengan menggunakan *Wu-Palmer*. Perhitungan *Wu-Palmer* ditunjukkan pada persamaan dibawah ini

$$Sim_{wwp}(c1, c2) = \frac{2xdepth(LCS(c1, c2))}{depth(c1) + depth(c2)} \quad (1)$$

dari setiap hasil dari persamaan tersebut akan dilakukan *metric similarity*. Dari hasil tersebut nantinya akan menjadi hasil yang menentukan apakah kedua kalimat tersebut berhubungan.

## 2.9 Kappa Statistic

Untuk skenario pengujian hasil deteksi ketidaksesuaian yang membandingkan hasil deteksi ketidaksesuaian antara kakas bantu dan ahli rekayasa perangkat lunak, hasil pengujian dinilai dari kesepakatan antara kakas bantu dan ahli rekayasa perangkat lunak melalui perhitungan *kappa statisticic*. Perhitungan *kappa statistic* diilustrasikan melalui contoh kasus berikut. Terdapat sebuah percobaan dimana pengamat A dan B mengelompokkan 100 data ke dalam dua kategori yang diberikan label “1” dan “2”. Hasil percobaan ini ditunjukkan pada tabel 2.2

**Tabel 2.3 Tabel contoh data dua pengamat**

| **Pengamat A** | **Pengamat B** | | |
|---|---|---|---|
| | "1" | "2" | Jumlah |
| **"1"** | 40 | 9 | 49 |
| **"2"** | 6 | 45 | 51 |
| **Jumlah** | 46 | 54 | 100 |

Perhitungan *kappa statistic* dimulai dengan menghitung *chance agreement*. perhitungan ditunjukkan pada persamaan (2) sebagai berikut

$$e(\gamma) = 2 \times P_1(1 - P_1) \tag{2}$$

Dimana $P_1$ Merepresentasikan perkiraan kemungkinan seorang pengamat (A atau B) mengelompokkan data ke kategori "1". Nilai $P_1$ Dihitung dengan persamaan (3) sebagai berikut

$$P_1 = \frac{(A1 + B1)}{2 \times N} \tag{3}$$

A1 dan B1 masing-masing adalah perkiraan kemungkinan seorang pengamat (A atau B) mengelompokkan data ke dalam kategori "1". Sedangkan *N* adalah jumlah data. Sehingga, nilai *chance agreement* dari hasil percobaan pada tabel 5.1 adalah

$$2\left(\frac{46+49}{2\times 100}\right)\left(1-\frac{46+49}{2\times 100}\right) = 0{,}49875$$

Nilai *kappa statistic* dapat dihitung setelah mendapatkan nilai *chance agreement* yang telah didapatkan dari perhitungan diatas.

Selanjutnya nilai *kappa statistic* dihitung dengan persamaan (4)

$$AC1 = \frac{p - e(\gamma)}{1 - e(\gamma)} \tag{4}$$

Dimana *P* adalah proporsi kesepakatan keseluruhan yang dihitung melalui persamaan

$$p = \frac{(A + D)}{N} \tag{5}$$

*A* adalah banyaknya data yang dikelompokkan dalam kategori "1" oleh kedua pengamat. *D* adalah banyaknya data yang dikelompokkan ke dalam kategori "2" oleh kedua pengamat. sehingga, hasil percobaan pada tabel 5.1 memilikki nilai *kappa statistic* :

$$\frac{(0{,}85 - 0{,}49875)}{(1 - 0{,}49875)} = 0{,}70074$$

dengan nilai *P* :

$$\frac{(40 + 45)}{100} = 0{,}85$$

Dari persamaan diatas didapatkan nilai *kappa statistic* 0,70074.

Untuk skenario pengujian ini, hasil pengujian dikelompokkan ke dalam dua kategori yaitu "sesuai" dan "tidak sesuai". triplet masuk

ke dalam kategori sesuai apabila triplet tersebut memilikki subyek, obyek dan predikat yang benar, sedangkan "tidak sesuai" apabila triplet terdapat subyek, obyek atau predikat yang salah. Adapun ahli yang dilibatkan adalah ahli rekayasa perangkat lunak. Ahli tersebut memilikki kemampuan dan pengalaman di bidang analisis dan perancangan sistem.

## 2.10 XML Metadata interchange (XMI)

*XML Metadata Interchange* (XMI) adalah sebuah standar pertukaran informasi *metadata* melalui *Extensible Markup Language* (XML). Standar tersebut berdasarkan *Meta Object Facility* (MOF), sebuah standar *meta-meta-model* dari *Object Management Group* (OMG). XMI menggabungkan sebuah *meta-meta-model* ke sebuah teks dalam for-mat XML. Untuk menghasilkan informasi *metadata* ke sebuah teks dalam format XML, XMI mendefinisikan serangkaian aturan *meta-model* berdasarkan MOF *meta-meta-model*. Salah satu contoh *meta-model* adalah UML. Sebuah UML didefinisikan sebagai sebuah model yang terdiri dari informasi *metadata* yang saling berkaitan karena mendeskripsikan informasi yang saling berhubungan, mendefinisikan sebuah rangkaian aturan yang mende-finisikan struktur dan konsistensi, dan memiliki makna dalam sebuah kerangka.

Kelebihan dari XMI adalah *portability*, memungkinkan sebuah model bisa digunakan di berbagai jenis perkakas rekayasa perangkat lunak. Hal tersebut terjadi karena XMI mengandung informasi *metadata* yang bisa direpresentasikan ke dalam sebuah diagram UML atau sebaliknya dari UML diagram ke XMI.Ada berbagai jenis aplikasi pemodelan UML yang mendukung konversi dari diagram UML ke dalam format XMI atau sebaliknya dari format XMI ke diagram UML. StarUML merupakan salah satu

aplikasi yang menyediakan fitur tersebut. Fitur ini bisa dijalankan dengan menambahkan XMI *extension* pada aplikasi StarUML. Dengan menggunakan bantuan *extension* tersebut, elemen yang ada pada XMI bisa menunjukkan apa dan bagai-mana komponen yang ada pada rancangan diagram UML disusun. Setiap elemen terdiri dari atribut dan nilai atribut Contoh Format XML ke diagram XML pada Tabel 2.3

**Tabel 2.4 Atribut xmi diagram urutan**

| **Elemen** | **Atribut** | **Keterangan** |
|---|---|---|
| Lifeline | Xmi:Id,Name,xmi:type, Represents | *xmi:id*,mendefinisikan id dari sebuah lifeline *name*,mendefinisikan nama dari sebuah lifeline *xmi:type*,mendefinisikan sebuah tipe dari lifeline *represents*,mendefenisikan sebuah refrensi UUID dari kelas yang bersangkutan |
| Message | Xmi:Id,name,xmi:type, sendEvent,receiveEvent | *Xmi:id*,mendefinisikan id dari sebuah *message* *Name*,merepresentasikan nama dari sebuah *message* *sendEvent* dan *receiveEvent*,merupakan refrensi UUID lifeline yang terlibat |
| Fragment | Xmi:id,covered | *Xmi:id*,mendefinisikan id sebuah fragment *Covered*,menunjukkan refrensi UUID lifeline |

**Tabel 2.3 Atribut xmi diagram urutan (*lanjutan*)**

| | | |
|---|---|---|
| Owned Attribut | Xmi:id,name,type | *Xmi:id*,mendefinisika id sebuah owned atribut<br>*Name*,mendefinisikan nama dari sebuah ownedAttribut<br>*Type*,mendefinisikan tipe dari sebuah owned attribut |

***(Halaman ini sengaja dikosongkan)***

# BAB III
# ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas analisis dan perancangan dari perangkat lunak yang hendak dibangun melalui pendekatan orientasi Objek. Analisis membahas pendefenisian kebutuhan sistem terhadap permasalahan yang diangkat di dalam tugas akhir. Selanjutnya perancangan membahas perancangan antarmuka dan proses yang ada di dalam perangkat lunak

## 3.1 Analisis Perangkat Lunak

Pada Sub-bab ini membahas analisis kebutuhan perangkat lunak. Analisis perangkat lunak secara garis besar berisi deskripsi umum dan kebutuhan perangkat lunak.

### 3.1.1 Deskripsi Umum Perangkat Lunak

Tugas Akhir ini berbentuk Perangkat Lunak untuk mendeteksi Ketidaksesuaian kodesumber terhadap diagram urutan . Perangkat lunak berupa kakas bantu berbasis desktop. Adapun pengguna dari kakas bantu ini nantinya adalah analis sistem, yang bertanggung jawab terhadap perubahan kode sumber terhadap diagram urutan.

Gambaran proses dari kakas bantu yang dibuat adalah sebagai berikut :

1. Pra-Proses: analis sistem memilih diagram urutan yang akan dipakai sebagai acuan sebagai pendeteksian ketidaksesuaian, analis memilih atau membuat diagram urutan dengan menggunakan aplikasi *Star-UML2*, kemudian analis sistem mengonversikan diagram urutan tersebut menjadi sebuah berkas dalam format ”*XMI*” menggunakan *extension StarUML2*. Selanjutnya analis memilih kode sumber yang berkaitan terhadap diagram urutan kemudian analis sistem mengkonversikan kode sumber menjadi berkas dalam format “*XMI*” menggunakan *xmltranslatorApp* [1]. Nantinya berkas

ekspor yang dihasilkan adalah berbentuk *XML*. Berkas kode sumber yang dihasilkan nantinya disebut dengan kumpulam dokumen kode sumber.

2. *Input*: diagram urutan yang memilikki format ”*.xmi*”, dan dokummen kode sumber yang berisi beberapa kode sumber berkaitan dengan diagram urutan dalam format “*.xml*”.
3. Proses: mengekstraksi diagram urutan yang terdiri dari *lifeline*, *message*, *fragment*, *ownedAttribute* dan *packaged element*. Dari hasil ekstraksi diagram urutan diolah untuk mendapatkan triplet yang membentuk subyek, predikat, dan obyek. Kemudian mengekstraksi folder yang terdiri banyak kode sumber, dari masing-masing file kode sumber dilakukan ekstraksi untuk mengambil *nama kelas* dari kode sumber, lalu *nama metode* yang terdapat di dalam kelas tersebut, kemudian *baris kode sumber* yang berisi dari pemanggilan metode didalam kelas obyek. Kemudian setelah mendapatkan ekstraksi dari diagram urutan dan kode sumber dilakukan proses matching yang dilakukan untuk mendeteksi setiap triplet dengan melakukan pendeteksian didalam kode sumber. Nantinya hasil dari proses ini akan membentuk sebuah penanda pada triplet, penanda itu akan menandakan ketidaksesuaian kode sumber terhadap diagram urutan. Triplet yang benar akan mendapatkan tanda centang (✓) pada subyek, predikat, dan obyeknya. Untuk triplet tidak sesuai akan mendapatkan tanda silang (✗) pada subyek, predikat, dan obyeknya.
4. Output: Berupa daftar triplet yang berisi subyek, predikat dan obyek dengan penanda centang (✓) atau penanda silang (✗) pada subyek, predikat, dan obyek.

### 3.1.2 Proses Bisnis

Proses bisnis yang terdapat pada kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan kode sumber:

| | | |
|---|---|---|
| **Nama** | : | Memeriksa ketidaksesuaian kode sumber terhadap diagram urutan |
| **Alur** | : | 1.Memasukkan file diagram urutan<br>2.Memilih folder kode sumber<br>3.Menekan Tombol proses<br>4.Mendapatkan hasil deteksi ketidaksesuaian diagram urutan |
| **Masukkan** | : | Diagram urutan dan folder yang berisi kode sumber yang telah berbentuk ".xmi" dan ".xml" |
| **Keluaran** | : | Hasil deteksi berupa triplet dengan penanda pada subyek, predikat, dan obyek nya |
| **Pelaku** | : | Analis sistem |

### 3.1.3 Fungsi Produk

Proses bisnis yang telah dijelaskan sebelumnya kemudian dipetakan ke dalam fungsi produk. Fungsi produk ini merupakan kebutuhan fungsional yang harus dipenuhi kakas bantu. Pemetaan tersebut dijabarkan oleh deskripsi berikut:

| | | |
|---|---|---|
| **Nama proses bisnis:** | | Memeriksa ketidaksesuaian diagram urutan terhadap kode sumber |
| **Fungsi Produk** | **:** | Kakas bantu dapat mendeteksi ketidaksesuaian diagram urutan terhadap kode sumber |
| **Asumsi** | **:** | File diagram urutan dan file kode sumber berbentuk format ".xmi" dan ".xml" |

Dari pemetaan diatas terdapat satu kebutuhan fungsional tersebut yaitu analis sistem dapat memeriksa ketidaksesuaian diagram urutan terhadap kode sumber.

### 3.1.4 Karakteristik Pengguna

Pengguna kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan kode sumber adalah analis sistem. Analis sistem memilikki tanggung jawab untuk memilih diagram urutan dan file kode sumber yang sesuai dengan diagram urutan yang sudah dipilih.

### 3.1.5 Batasan

Adapun batasan dari kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan kode sumber. Batasan terkait metode dan teknologi yang digunakan saat implementasi perangkat lunak. Batasan tersebut antara lain :

1. Kakas bantu ini dibuat dengan menggunakan bahasa Java
2. Kakas bantu ini hanya mendeteksi kode sumber dengan berbasis bahasa java
3. Kakas bantu hanya bisa membaca diagram urutan dengan format ".xmi" dengan menggunakan plugin aplikasi Star-UML
4. Kakas bantu hanya bisa membaca kode sumber dengan format ".xml" dengan menggunakan *xmltranslatorapp*
5. Kakas bantu hanya bisa mendeteksi *message* pada diagram urutan
6. Kakas bantu berbasis desktop

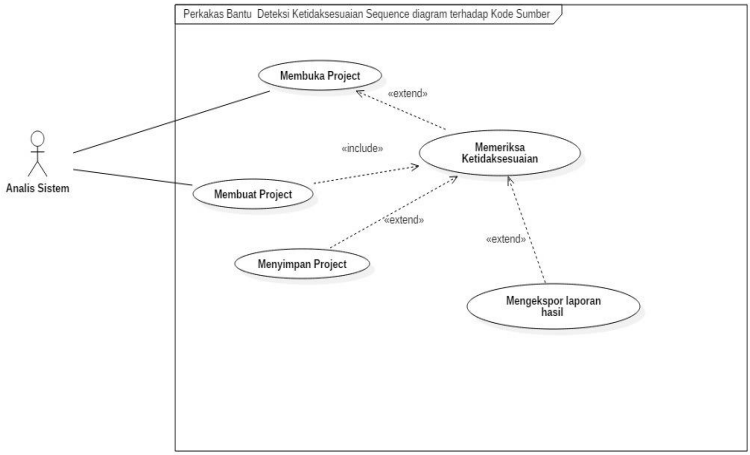
### 3.1.6 Kebutuhan non-fungsional

Di dalam analisis ini, tidak ditemukan kebutuhan non-fungsional di dalam kakas bantu ini

### 3.1.7 Diagram Kasus Penggunaan (Use Case Diagram)

Kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan kode sumber memilikki dua kasus penggunaan dasar (*base use case*) yaitu membuka proyek, membuat proyek dan empat

kasus sub kasus penggunaan yaitu memeriksa ketidaksesuaian, menyimpan proyek, dan mengeskpor hasil laporan. Fungsionalitas tersebut diterjemahkan ke dalam Diagram kasus penggunaan. Kasus penggunaan merepresentasikan kebutuhan utama (fungsionalitas) yang dibutuhkan oleh aktor. Diagram kasus penggunaan dari kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan terhadap kode sumber ditunjukkan pada Gambar 3.1 menunjukkan ada satu kasus penggunaan utama dari kakas bantu yaitu memeriksa ketidaksesuaian dan 4 buah sub kasus penggunaan. Penjelasan lebih dalam dari kasus penggunaan ini dijelaskan pada bagian berikutnya.




**Gambar 3.1 Diagram kasus penggunaan kakas bantu**

Pada kasus penggunaan diatas didapatkan kasus penggunaan dasar adalah “memeriksa ketidaksesuaian”. Pada usecase diatas didapatkan kasus penggunaan “membuat proyek” memilikki relasi yang dinotasikan dengan notasi *include*. Notasi relasi *include* menjelaskan kasus penggunaan “membuat proyek” harus dilakukan terlebih dahulu untuk aktor agar dapat mencapai atau

melakukan kasus penggunaan “memeriksa ketidaksesuaian diagram urutan”. Lalu selanjutnya kasus penggunaan “membuka proyek” dan “mendeteksi ketidaksesuaian diagram urutan” memilikki relasi yang dinotasikan dengan extend. Notasi relasi extend menjelaskan bahwa kasus penggunaan “membuka project” adalah opsional yang berarti kasus penggunaan ini memungkinkan untuk dilakukan ataupun tidak dilakukan. Lalu untuk kasus penggunaan selanjutnya yaitu “menyimpan proyek” dan “mengekspor hasil laporan” memilikki relasi yang sama terhadap kasus penggunaan “membuka project” keduanya memilikki relasi *extend* yang menjelaskan bahwa kedua kasus penggunaan ini optional yang berarti kaus penggunaan ini memungkinkan untuk dilakukan ataupun tidak dilakukan.

Penjelasan alur dasar setiap kasus penggunaan akan dijelaskan sebagai berikut:

1. Kasus Penggunaan “Memeriksa ketidaksesuaian kode sumber”

Menjelaskan bagaimana analis sistem melakukan pemeriksaan ketidaksesuaian dokumen rancangan. Kasus penggunaan ini memilikki relasi include dengan kasus penggunaan “membuat proyek”. Alur dasar dari kasus penggunaan “Memeriksa ketidaksesuian” dijabarkan sebagai berikut:

1. Analis sistem membuka kakas bantu
2. Analis sistem membuat project (include kasus penggunaan “membuat proyek”)
3. Sistem menampilkan halaman utama kakas bantu
4. Analis sistem memilih memasukkan berkas (*file*) diagram urutan
5. Kakas bantu menampilkan jendela untuk pemilihan berkas

6. Analis memilih diagram urutan sesuai dengan *file* ekstensi yang sudah ditentukan
7. Sistem menampilkan hasil ekstraksi diagram urutan pada *textarea* diagram urutan
8. Analis menekan tombol untuk memilih folder direktory kode sumber
9. Kakas bantu menampilkan direktori *folder*
10. Analis memilih direktori tempat kumpulan kode sumber dengan ekstensi ".xml" yang hendak diproses
11. Kakas bantu menampilkan hasil proses ekstraksi didalam text area kode sumber
12. Analis memilih proses
13. Kakas bantu menampilkan hasil proses analisis

2. Kasus Penggunaan "Membuat Proyek"

Menjelaskan bagaimana analis sistem Membuat proyek. Kasus penggunaan ini memilikki relasi dengan notasi *include* terhadap kasus penggunaan "Mendeteksi ketidaksesuaian". Alur dasar kasus penggunaan ini dijelaskan sebagai berikut:

1. Analis sistem membuka kakas bantu pendeteksian
2. Sistem menampilkan halaman pertama pada kakas bantu
3. Analis sistem memilih membuat proyek pada menubar kakas bantu
4. Sistem menampilkan halaman utama kakas bantu

3. Kasus Penggunaan "Membuka Proyek"

Menjelaskan bagaimana analis sistem membuka proyek. Kasus penggunaan ini memilikki relasi dengan notasi *extend* terhadap kasus penggunaan "mendeteksi ketidaksesuaian". Alur dasar kasus penggunaan ini terjadi pada nomer 1 pada alur dasar "mendeteksi ketidaksesuaian". Alur dijelaskan sebagai berikut:

1.1 Analis sistem membuka kakas bantu pendeteksian

2.1 Sistem menampilkan halaman pertama kakas bantu
3.1 Analis sistem menekan "membuka proyek" pada menu bar
4.1 Sistem menampilkan jendela pemilihan file proyek
5.1 Analis sistem memilih file proyek yang sudah disimpan sebelumnya

4. Kasus Penggunaan "Menyimpan Proyek"

Menjelaskan bagaimana analis sistem menyimpan proyek. Kasus penggunan ini memilikki relasi dengan notasi *extend* terhadap kasus penggunaan "Mendeteksi ketidaksesuaian". Alur dasar kasus penggunaan ini terjadi pada nomer 10 pada alur dasar "mendeteksi ketidaksesuaian". Penjelasan alur dijelaskan sebagai berikut:

1.10 Analis sistem menekan tombol "simpan"
2.10 Sistem menampilkan jendela direktori folder
3.10 Analis sistem memilih tempat penyimpanan
4.10 Analis menekan tombol save pada jendela direktori folder

5. Kasus Penggunaan "Mengekspor hasil laporan"

Menjelaskan bagaimana analis sistem mengekspor hasil laporan. Kasus penggunaan ini memilikki relasi dengan notasi *extend* terhadap kasus penggunaan "Mendeteksi ketidaksesuaian". Alur dasar kasus pengganaan ini terjadi pada nomer 10 pada alur dasar "Mendeteksi ketidaksesuaian". Penjelasan alur dijelaskan sebagai berikut:

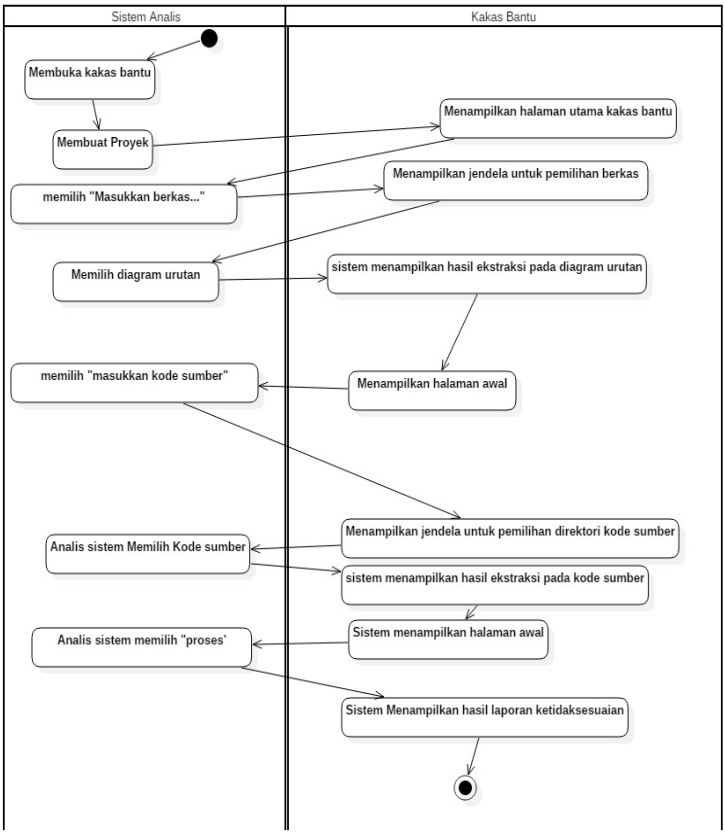
1.10 Analis menekan tombol "export pdf".
2.10 Sistem menampilkan jendela direktori folder.
3.10 Analis sistem memilih tempat penyimpanan.
4.10 Sistem menampilkan hasil analisis pada folder tujuan.

### 3.1.8 Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas adalah diagram yang digunakan untuk menggambarkan proses bisnis atau urutan aktivitas dalam sebuah proses. Diagram ini juga bisa digunakan untuk menjelaskan langkah-langkah bagaimana suatu kasus penggunaan dijalankan. Alur dasar yang telah dijabarkan pada subbab penjelasan kasus penggunaan sebelumnya kemudian direpresentasikan ke dalam diagram aktivitas. Diagram tersebut diantaranya sebagai berikut:

#### 3.1.8.1 Diagram Aktivitas : Memeriksa ketidaksesuaian kode sumber

Diagram aktivitas untuk kasus penggunaan "memeriksa ketidaksesuaian" menggambarkan langkah-langkah untuk melakukan kasus penggunaan memeriksa ketidaksesuaian. Proses langkah-langkah tersebut telah dijabarkan pada bab 3.1.7 pada bagian 1. Dari langkah-langkah tersebut digambarkan menjadi sebuah diagram aktivitas yang digambarkan pada gambar 3.2.

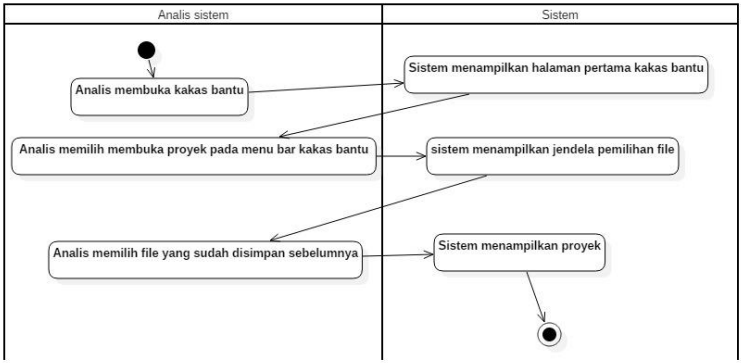



**Gambar 3.2 Gambar diagram aktivitas memeriksa ketidaksesuaian**

### 3.1.8.2 Diagram Aktivitas : Membuka Proyek

Diagram Aktivitas untuk kasus penggunaan "membuka proyek" menggambarkan langkah-langkah untuk analis dalam mencapai kasus penggunaan membuka proyek. Proses langkah-langkah tersebut telah dijabarkan pada bab 3.1.7 bagian 3. Setelah itu

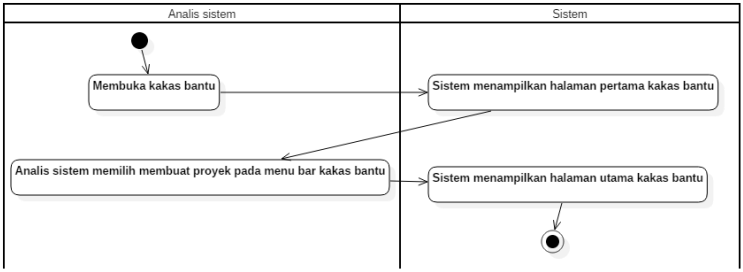
langkah-langkah tersebut digambarkan menjadi diagram aktivitas yang ditunjukkan oleh Gambar 3.3




**Gambar 3.3 Diagram aktivitas membuka proyek**

### 3.1.8.3 Diagram Aktivitas : membuat proyek

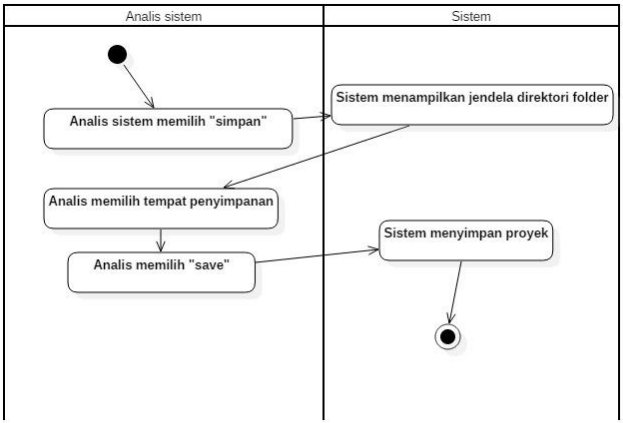
Diagram Aktivitas untuk kasus penggunaan "membuat proyek" menggambarkan langkah-langkah untuk melakukan kasus penggunaan "membuat proyek". Proses langkah-langkah tersebut telah dijabarkan pada bab 3.1.7 pada bagian ke 2. Dari langkah-langkah tersebut digambarkan menjadi sebuah diagram aktivitas yang ditunjukkan oleh Gambar 3.4




**Gambar 3.4 diagram aktivitas membuat proyek**

### 3.1.8.4 Diagram Aktivitas : menyimpan proyek

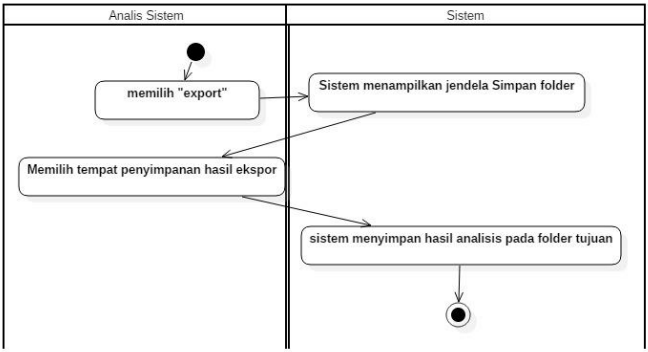
Diagaram Aktivitas untuk kasus penggunaan “menyimpan proyek” menggambarkan langkah-langkah analis untuk melakukan kasus penggunaan “menyimpan proyek”. Proses langkah-langkah tersebut telah dijelaskan pada bab 3.1.7 bagian 4. Dari langkah-langkah tersebut dibentuk diagram aktivitas yang ditujukkan pada Gambar 3.5.




**Gambar 3.5 Diagram aktivitas menyimpan proyek**

### 3.1.8.5 Diagram Aktivitas : Mengekspor Hasil laporan

Diagaram Aktivitas untuk kasus penggunaan “mengekspor hasil proyek” menggambarkan langkah-langkah untuk melakukan kasus penggunaan “mengespor hasil laporan”. Proses langkah-langkah tersebut telah dijabarkan pada bab 3.1.7 bagian ke 5. Dari langkah-langkah tersebut digambarkan menjadi sebuah diagram aktivitas yang ditunjukkan oleh Gambar 3.6




**Gambar 3.6 Gambar diagram aktivitas mengekspor hasil laporan**

### 3.1.9 Kelas Analisis

Diagram kasus penggunaan yang telah digambarkan sebelumnya akan menjadi bahan model analisis. Model analisis diperlukan untuk menemukan kelas-kelas analisis pada setiap kasus penggunaan. Selain itu, model analisis menghasilkan spesifikasi kebutuhan yang lebih tepat dan lengkap. Penjelasan alur setiap kasus penggunaan dilengkapi terlebih dahulu sebelum melakukan model analisis. Sesuai gambar 3.1, terdapat satu kasus penggunaan yaitu memeriksa ketidaksesuian diagram urutan. Proses model analisis kasus penggunaan tersebut dijelaskan secara detail pada sub bab berikut

#### 3.1.9.1 Kelas Analisis Memeriksa Ketidaksesuaian Diagram urutan

Kasus penggunaan ini dilakukan oleh seorang aktor, yaitu analis sistem. Setelah analis sistem mendapatkan diagram urutan dan kode sumber yang sesuai dengan diagram urutan. Lalu setelah itu analis sistem melakukan pendeteksian untuk mengetahui apakah diagram urutan sesuai apa belum dengan kode sumber. Model

analisis kasus penggunaan “memeriksa ketidaksesuaian diagram urutan” dilakukan melalui langkah-langkah berikut ini:

1. Melengkapi penjelasan kasus penggunaan

Berdasarkan alur dasar dari kasus penggunaan “memeriksa ketidaksesuaian diagram urutan”, kasus penggunaan ini dimulai saat analis sistem menekan tombol “memasukkan berkas diagram urutan”. Layar utama kakas bantu mengirimkan perintah untuk menampilkan jendela “direktori file”. Jendela tersebut digunakan untuk memilih lokasi dimana berkas diagram urutan disimpan sesuai dengan ekstensi file “.xmi”. Setelah analis memilih berkas diagram urutan, kakas bantu menampilkan *path file* di dalam bidang teks (*textfield*) pada area diagram urutan kakas bantu, dan menampilkan “hasil ekstraksi diagram urutan” yang berupa triplet yang terdiri dari subyek, predikat, obyek. Triplet terbentuk dari *lifeline*, *message*, *fragment*, dan *owned attribute*. Setelah kakas bantu menampilkan hasil ekstraksi diagram urutan, analis menekan tombol “memilih folder kode sumber”. Layar utama kakas bantu mengirimkan perintah untuk menampilkan jendela “direktori folder”. Jendela tersebut digunakan untuk memilih lokasi dimana direktori terdapat, direktori itu tersebut berisi banyak kode sumber yang sudah berbentuk format “.xml” setelah analis memilih direktori tersebut, kakas bantu menampilkan path folder di dalam textfield pada area kakas bantu, dan menampilkan “hasil ekstraksi kode sumber” yang berupa nama kelas, nama metode, dan nama pemanggilan kode sumber yang terbentuk dari pemanggilan metode dari kelas obyek yang terdapat di dalam sebuah metode. Setelah kakas bantu menampilkan hasil ekstraksi kode sumber. Analis “menekan tombol proses”,kakas bantu mengirimkan perintah untuk melakukan pendeteksian ketidaksesuaian. Pendeteksian dilakukan dengan melakukan pengecekkan pada

setiap triplet yang ditemukan pada diagram urutan pada hasil ekstraksi kode sumber dengan menggunakan metode *string matching*. Pendeteksian pertama dilakukan dengan mencari Subyek apakah sebuah aktor, lalu menemukan subyek di dalam kelas dari hasil ekstraksi kode sumber, setelah menemukan subyek terdapat di dalam kelas dan bukan aktor, pendeteksian dilanjutkan dengan mengecek apakah terdapat pemanggilan kode sumber yang terbentuk dari predikat dengan obyek, apabila ditemukan pemanggilan kode sumber di dalam kelas subyek maka subyek mendapatkan tanda benar. Setelah Pendeteksian subyek pada kakas bantu sukses,pendeteksian berikutnya menemukan obyek didalam kelas dari hasil ekstraksi kode sumber dan obyek bukan merupakan aktor,maka obyek mendapatkan tanda benar, lalu setelah itu pendeteksian dilanjutkan dengan mendeteksi predikat,pendeteksian predikat dilakukan dengan mengecek isi metode didalam kode sumber yang sudah di ekstraksi, selanjutnya mendeteksi apakah predikat terdapat di dalam kelas obyek sebelumnya. Setelah berhasil maka apabila ketiga bagian triplet itu benar,maka triplet dapat dikatan triplet benar. Apabila terdapat salah pada salah satu triplet maka dapat dikatakan bahwa triplet itu salah dan tidak sesuai pada kode sumber. Nantinya setelah setiap triplet berhasil di deteksi, kakas bantu menampilkan hasil deteksi berupa banyaknya triplet dengan penanda benar atau salah pada setiap subyek, predikat, dan obyek.

2. Menemukan kandidat kelas dari perilaku kasus penggunaan.

Untuk setiap alur kasus penggunaan yang telah dilengkapi penjelasaannya kemudian dianalisis perilakunya untuk mendapatkan kelas. Kandidat kelas bisa ditemukan melalui pemilihan kata benda pada alur tersebut. Tabel 3.1 menunjukkan kandidat kelas hasil analisis melalui pendekatan kata benda.

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan"**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 1 | Analis- sistem | Peran yang memilikki tanggung jawab untuk pengembangan dalam sebuah Perangkat lunak | Bukan | Analis sistem merupakan pengguna dari kakas bantu ini. |
| 2 | Tombol "Masukkan berkas diagram urutan" | Tombol yang digunakan untuk memasukkan berkas diagram urutan | Bukan | Tombol ini merupakan salah satu komponen penyususun (atribut) dari layar utama kakas bantu |
| 3 | Tombol "pilih folder kode sumber" | Tombol yang digunakan untuk memilih direktori folder yang berisi kode sumber | Bukan | Tombol ini merupakan salah satu komponen penyusun (atribut) dari layar utama kakas bantu |

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan" (*lanjutan*)**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 4 | Layar utama kakas bantu | Antar muka kakas bantu yang menghubungkan analis sistem dengan kakas bantu (tempat proses bisnis berlangsung) | Ya | Layar kakas bantu terdiri dari komponen seperti , tombol "masukkan berkas diagram urutan" , lalu tombol "pilih folder kode sumber" dan tombol "proses" |
| 5 | Diagram urutan | Merupakan diagram yang menggambarkan interaksi antar object-object / lifeline-lifeline melalui urutan pesan (*message*) | Bukan | Diagram urutan merupakan elemen yang digunakan untuk pemeriksaan ketidaksesuaian. Elemen diagram ini disimpan dalam bentuk triplet |
| 6 | Jendela "direktori file" | Jendela untuk memilih berkas diagram urutan (*sequence diagram*) yang disimpan dalam lokasi tertentu | Ya | Jendela ini merupakan antarmuka yang menangani pemilihan berkas, seperti menampilkan lokasi dimana berkas disimpan, mengatur ekstensi berkas yang dipilih dan memilih berkas |

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan" (*lanjutan*)**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 7 | Bidang teks(*textfield*) | Area yang digunakan untuk menampilkan direktori file ataupun folder yang sudah dipilih oleh analis | Bukan | Bidang teks(*textfield*) merupakan salah satu komponen penyusun (*atribut*) dari layar utama kakas bantu |
| 8 | Ekstraksi diagram urutan | Proses untuk melakukan ekstraksi diagram urutan yang sudah dipilih. Esktraksi dilakukan dengan tujuan untuk membentuk sebuah triplet yang terdiri dari subyek, predikat,obyek | Ya | Proses ini bertujuan untuk membentuk triplet yang menjadikan dasar untuk pendeteksian ketidaksesuaian,tripet terdiri dari subyek,predikat,dan obyek. Triplet nantinya dibentuk dari hasil ekstraksi elemen lifeline, message , fragment yang terdapat didalam diagram urutan. |
| 9 | Kode sumber | Merupakan isi barisan kode yang sudah dibentuk sedimikian rupa. | Bukan | Kode sumber merupakan elemen yang akan dideteksi terhadap diagram urutan untuk mencari ketidaksesuaian pada diagram urutan. |

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan" (*lanjutan*)**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 10 | Ekstraksi kode sumber | Proses untuk melakukan ekstraksi kode sumber yang sudah dipilih | Ya | Proses ini bertujuan untuk membentuk data berisikan kelas, nama metode,dan baris kode sumber di dalam metode sebuah kelas. |
| 11 | Triplet | Merupakan hasil yang didapatkan dari ekstraksi diagram urutan | Ya | Triplet merupakan hasil dari ekstraksi diagram urutan yang nantinya dijadikan pedoman untuk melakukan pendeteksian ketidaksesuaian, triplet ini berisikan Subyek, Predikat, dan Obyek |
| 12 | Nama kelas | Merupakan hasil yang didapatkan dari ekstraksi kode sumber | Ya | Nama kelas merupakan hasil ekstraksi kode sumber yang akan digunakan sebagai pembanding pada saat melakukan pendeteksi ketidaksesuain |

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan" (*lanjutan*)**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 13 | Nama metode | Merupakan hasil yang didapatkan dari ekstraksi kode sumber (*Source code*) | Ya | Nama metode merupakan hasil ekstraksi kode sumber yang digunakan sebagai pembanding pada saat melakukan pendeteksi ketidaksesuaian |
| 14 | Pemanggilan kode sumber | Merupakan hasil yang didapatkan dari ekstraksi kode sumber (*Source code*) | Ya | Pemanggilan kode sumber merupakan hasil ekstraksi kode sumber yang digunakan sebagai pembanding pada saat melakukan pendeteksi ketidaksesuaian |

**Tabel 3.1 Kandidat kelas kasus penggunaan "memeriksa ketidaksesuaian diagram urutan" (*lanjutan*)**

| No | Teks | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| 15 | Pendeteksian ketidaksesuaian | Proses untuk memeriksa ketidaksesuaian diagram urutan (*Sequence diagram*) dengan kode sumber | Ya | Pendeteksian ketidaksesuaian mengatur pendeteksian mulai dari pengecekan pada setiap triplet ke dalam kode sumber , sampai pada memproses triplet untuk mendapatkan hasil pendeteksian |
| 16 | Pencocokan kata (*String Matching*) | Proses mendeteksi kesamaan kata pada diagram urutan dengan kode sumber | Ya | Metode yang digunakan untuk melakukan pendeteksian kata pada diagram urutan dan kode sumber |

3. Membagi perilaku kasus penggunaan ke kelas analisis

Untuk setiap Kandidat-kandidat kelas yang telah terpilih kemudian dijabarkan ke dalam tiga stereotype kelas. Stereotype tersebut, antara lain kelas boundary, kelas control, dan kelas entity. Kelas boundary merupakan kelas yang menghubungkan antara entitas diluar sistem dengan didalam sistem. Dengan kata lain, kelas ini berupa form, window sistem, atau interface untuk sistem lain.

Kelas control adalah kelas yang mengendalikan aliran logika sebuah kasus penggunaan dan mengoordinasikan objek lain. Kelas entity adalah kelas yang menyimpan informasi atau data yang digunakan oleh sistem dalam jangka waktu yang lama. Biasanya, informasi tersebut disimpan ke dalam database atau file. Berdasarkan Tabel 3.1, pemetaan kandidat kelas yang terpilih ke dalam nama kelas dan jenis kelas pada kakas bantu dijelaskan secara detail pada Tabel 3.2 sebagai berikut :

**Tabel 3.2 Tabel kandidat kelas**

| Teks/Nama kelas | Deskripsi | Jenis |
|---|---|---|
| **Layar utama kakas bantu** /*MainPage* | Merepresentasikan objek antarmuka dari kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan (*sequence diagram*) dengan kode sumber |  |
| **Jendela "Masukkan berkas diagram urutan"** / *FileChooser* | Merepresentasikan antar muka untuk memasukkan diagram urutan yang akan dideteksi ketidaksesuaiannya |  |
| **-**/*FileController* | Merepresentasikan antar muka untuk memilih folder yang akan dijadikan pembanding untuk mendeteksi ketidaksesuaian |  |
| **Ekstraksi diagram urutan**/*FileParser* | Merepresentasikan objek yang mengekstraksi informasi dari berkas diagram urutan yang sudah dipilih |  |

**Tabel 3.2 Tabel kandidat kelas (*lanjutan*)**

| Teks/Nama kelas | Deskripsi | Jenis |
|---|---|---|
| **Ekstraksi kode sumber**/*CodeParserStack* | Merepresentasikan objek yang mengekstraksi informasi dari berkas-berkas kode sumber dari direktory yang sudah dipilih |  |
| **Pendeteksian ketidaksesuaian** /*Matching* | Merepresentasikan objek yang mengatur pemeriksaan ketidaksesuaian diagram urutan dengan kode sumber |  |
| **Triplet**/*Triplet* | Merepresentasikan objek yang menyimpan hasil ekstraksi dari diagram urutan (*Sequence diagram*) |  |
| **Nama Kelas** /*Kelas* | Merepresentasikan objek yang menyimpan hasil ekstraksi kelas dari kode sumber |  |
| **Nama Metode**/*Metode* | Merepresentasikan objek yang menyimpan hasil ekstraksi metode dari kode sumber |  |
| **Baris kode sumber**/*BarisKodeSumber* | Merepresentasikan objek yang menyimpan hasil ekstraksi kode sumber dari kode sumber |  |

4. Mendeskripsikan tanggung jawab kelas

Untuk setiap hasil kelas analisis yang telah dipetakan sebelumnya pada Tabel 3.2 kemudian dideskripsikan tanggung jawabnya. Tanggung jawab kelas mendeskripsikan apa yang dilakukan oleh objek tersebut. Tanggung jawab kelas dapat diidentifikasi melalui kata kerja yang ada pada alur kasus penggunaan (*Use case*). Hasil identifikasi tersebut selanjutnya dipetakan ke dalam *method-method*. Hasil identifikasi kata kerja yang sudah dipetakan ke dalam method untuk setiap hasil kelas analisis dijabarkan oleh Tabel 3.3.

**Tabel 3.3 Tanggung jawab kelas**

| Nama | Notasi | Tanggung Jawab |
|---|---|---|
| *MainPage* |  | Mengirimkan perintah untuk menampilkan jendela "masukkan berkas diagram urutan" dan "memilih direktori kode sumber" serta untuk melakukan pemeriksaan ketidaksesuaian dokumen rancangan |
| *FileChooser* |  | Mengirimkan penanda jika sebuah berkas diagram urutan sudah dipilih dan mengirimkan perintah untuk memasukkan berkas yang dipilih tersebut ke dalam kakas bantu |
| *FileController* |  | Mengirimkan perintah untuk memasukkan berkas ke dalam kakas bantu. |
| *FileParser* |  | Melakukan ekstraksi diagram urutan dengan mengambil elemen lifeline, message,fragment , dan owned attribute dari xml diagram urutan |

**Tabel 3.3 Tanggung jawab kelas (*lanjutan)***

| | | |
|---|---|---|
| *CodeParserStack* |  | Melakukan ekstraksi kode sumber dengan mengambil elemen nama kelas , nama metode di dalam sebuah kelas, serta mengambil kode sumber yang terdapat di dalam sebuah metode. |
| *Matching* |  | Melakukan pendeteksian dengan menggunakan persamaan kata dengan mendeteksi setiap triplet terhadap kode sumber, pendeteksian dilakukan dengan mendeteksi melakukan pendeteksian Subyek , Predikat , dan Obyek. |
| *Triplet* |  | Menyimpan hasil ekstraksi diagram urutan dengan membentuk Subyek , Predikat, dan Obyek |
| *Kelas* |  | Menyimpan nama kelas dari hasil ekstraksi kode sumber |
| *Metode* |  | Menyimpan nama Metode dari hasil ekstraksi kode sumber |
| *BarisKodeSumber* |  | Menyimpan baris kode sumber dari hasil ekstraksi kode sumber |

5. Mendeskripsikan atribut dan hubungan kelas

Untuk setiap kelas-kelas yang telah dijelaskan tanggung jawabnya melalui *method-method* kemudian dijabarkan kembali untuk mendiskripsikan atribut-atribut dan hubungannya dengan kelas lain. Atribut kelas digunakan untuk menyimpan data atau informasi dari sebuah kelas, sedangkan hubungan kelas menunjukkan komunikasi antara kelas-kelas yang memilikki tujuan untuk menjalankan kasus penggunaan. Komunikasi terjadi saat saat suatu kelas mengirimkan atau menerima metode dari kelas lain. Penjabaran dari atribut dan hubungan kelas ditunjukkan oleh Tabel 3.4.

**Tabel 3.4 Tabel Hubungan Kelas**

| **Nama** | **Atribut** | **Hubungan** |
|---|---|---|
| MainPage | Teks area menampilkan *filepath* , tombol "memasukkan berkas diagram urutan" ,"memilih folder kode sumber", dan tombol "proses" dan area menampilkan hasil pendeteksian | *File Chooser* |
| File Chooser | Tombol "open" | *FileController* |
| FileController | - | *FileParser* |
| FileParser | - | *Triplet,Project,MainPage* |

**Tabel 3.4 Tabel Hubungan Kelas (*lanjutan)***

| | | |
|---|---|---|
| MainPage | Teks area menampilkan *filepath* , tombol "memasukkan berkas diagram urutan" ,"memilih folder kode sumber", dan tombol "proses" dan area menampilkan hasil pendeteksian | *FileController* |
| FileController | - | *CodeParserStack* |
| CodeParserStack | - | *Kelas,Metode,BarisKode ,MainPage* |
| MainPage | Teks area menampilkan *filepath* , tombol "memasukkan berkas diagram urutan" ,"memilih folder kode sumber", dan tombol "proses" dan area menampilkan hasil pendeteksian | *FileController* |
| FileController | - | *Matching* |
| Matching | - | *MainPage* |

### 3.1.10 Diagram Urutan (Sequence Diagram)

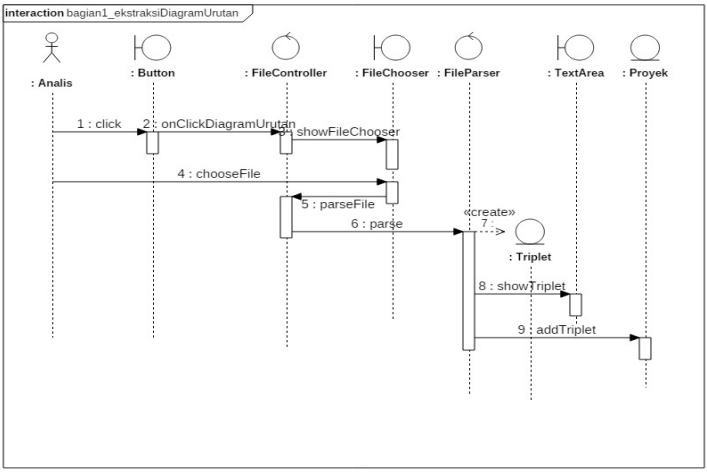
Dari analisis yang telah dilakukan pada sub bab sebelumnya, kelas kelas yang telah di dapatkan memilikki hubungan satu sama lain yang berinteraksi melalui serangkaian method. Bentuk interaksi tersebut digambarkan ke dalam sebuah diagram yaitu diagram

urutan. diagram urutan adalah diagram yang merealisasikan sebuah kasus penggunaan melalui interaksi dari serangkaian objek-objek. Karena terdapat beberapa kasus yang telah dianalisis perilakunya, gambar masing-masing diagram urutan akan dibagi menjadi beberapa sub-bab sesuai dengan nama kelas.

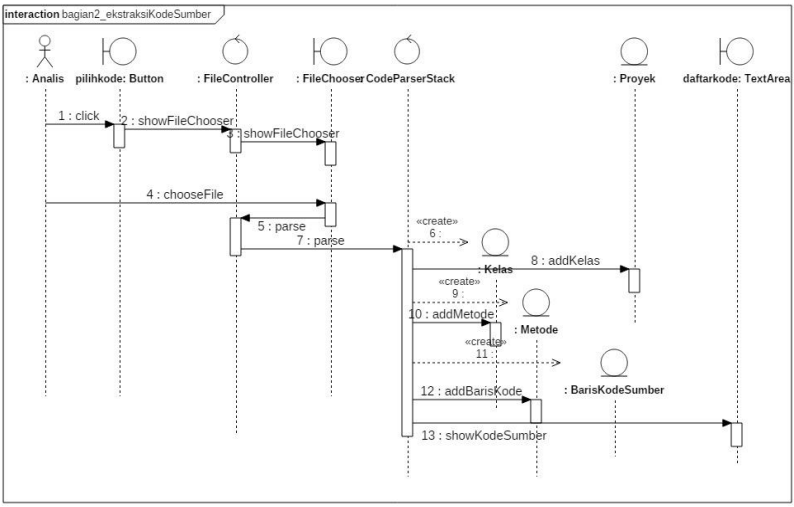
### 3.1.10.1 Diagram Urutan : Mendeteksi ketidaksesuaian

Dari analisis yang telah dilakukan sebelumnya didapatkan Hubungan kelas yang digambarkan pada Tabel 3.4 dibentuk menjadi sebuah interaksi yang berisikan nama kelas, dan interaksi metode yang digambarkan melalui pesan, diagram urutan ini nantinya menggambarkan fungsi utama dari kakas bantu ini. Kelas-kelas yang terlibat dalam interaksi sesuai dengan hasil analisis sebelumnya. Dikarenakan gambar diagram urutan terlalu besar, maka gambar diagram urutan akan dipecah menjadi 3 bagian,yaitu:

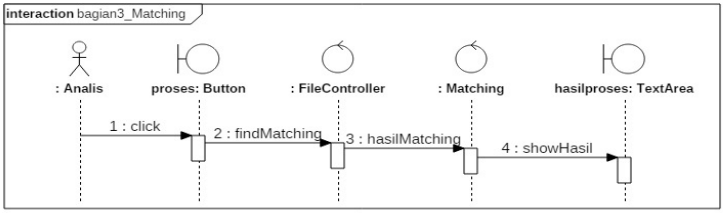
1. Bagian ekstraksi diagram urutan



2. Bagian ekstraksi kode sumber

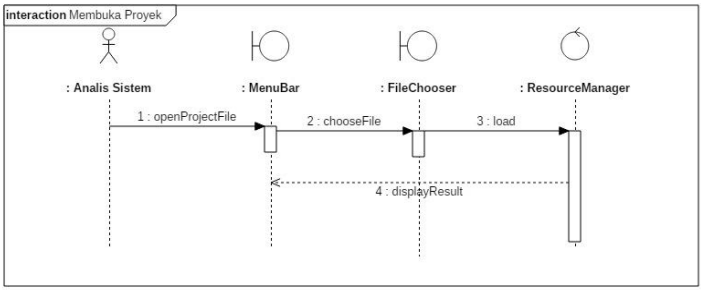



3. Bagian mendeteksi ketidaksesuaian




### 3.1.10.2 Diagram Urutan: Membuka Proyek

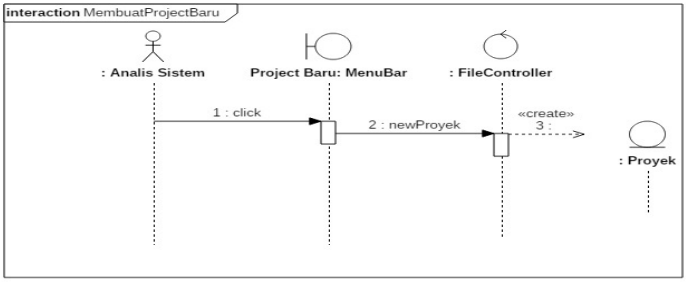
kelas-kelas yang terlibat dalam interaksi sesuai dengan hasil analisis sebelumnya. Didalam diagram urutan membuka proyek, terdapat 3 buah kelas yaitu, *MenuBar* yang merupakan salah satu komponenen dari Layar utama kakas bantu, lalu *FileChooser* dan terdapat *ResourceManager*. Nantinya masing-masing kelas ini akan berinteraksi dan digambarkan oleh gambar 3.7.




**Gambar 3.7 Gambar diagram urutan membuka proyek**

### 3.1.10.3 Diagram Urutan: Membuat Proyek

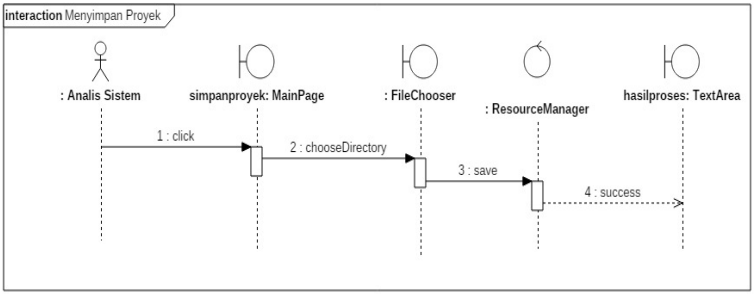
Gambar diagram urutan dari kasus "Membuat proyek" ditunjukkan oleh Gambar 3.8. Kelas-kelas yang terlibat dalam interaksi sesuai dengan hasil analisis sebelumnya. Dalam diagram urutan ini menunjukkan bagaimana proses analis sistem membuat proyek baru dalam kakas bantu ini. Proses dimulai ketika analis sistem menekan project baru pada menubar, setelah itu kakas bantu akan menampilkan dan membuat sebuah proyek baru untuk melakukan pendeteksian.




**Gambar 3.8 Diagram urutan membuat proyek**

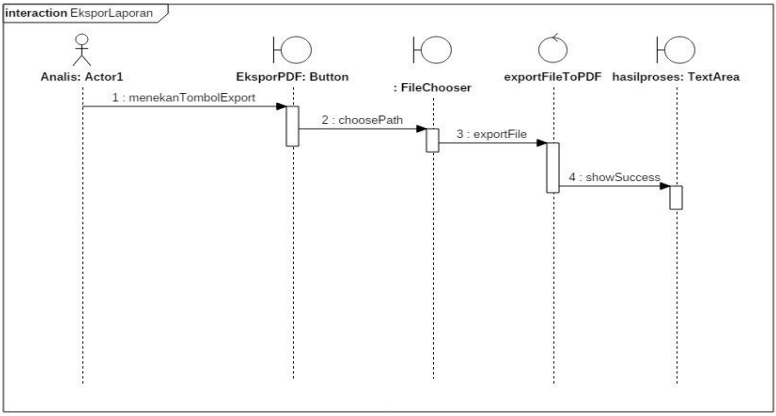
### 3.1.10.4 Diagram Urutan: Menyimpan Proyek

Gambar diagram urutan dari kasus "Menyimpan proyek" ditunjukkan oleh Gambar 3.9. Kelas-kelas yang terlibat dalam interaksi sesuai dengan hasil analisis sebelumnya.




**Gambar 3.9 Diagram urutan menyimpan proyek**

### 3.1.10.5 Diagram Urutan: Mengekspor Proyek




**Gambar 3.10 Diagram urutan Mengekspor proyek**

Gambar diagram urutan dari kasus “mengekspor proyek” ditunjukkan oleh gambar 3.10 kelas-kelas yang terlibat dalam interaksi sesuai dengan hasil analisis sebelumnya.

## 3.2 Perancangan Perangkat Lunak

Bab ini akan dibahas perancangan perangkat lunak dari Tugas Akhir yang berjudul “Pembuatan kakas bantu Deteksi Ketidaksesuaian kode sumber dengan diagram urutan”.

### 3.2.1 Perancangan Antar muka

Subbab ini membahas spesifikasi rancangan antar muka dari kakas bantu. Spesifikasi rancangan antar muka meliputi deskripsi atribut apa saja yang ada pada rancangan antar muka dan desain antar muka.“Kakas bantu untuk mendeteksi ketidaksesuaian kode sumber dengan diagram urutan” memilikki satu antar muka. Antar muka tersebut dijelaskan pada bahasan berikut ini

#### 3.2.1.1 Antar Muka Utama

Antar muka Utama adalah layar yang muncul ketika analis sistem membuat proyek pendeteksian. Kakas bantu ini dan layar ini akan menjadi layar inti di dalam kakas bantu ini. Antar muka utama terdiri dari komponen-komponen yang mendukung kebutuhan utama analis sistem. Komponen tersebut, antara lain komponen untuk memasukkan berkas, memilih folder dan menampilkan hasil pendeteksian. Penjelasan masing-masing komponen dijelaskan pada Tabel 3.5

**Tabel 3.5 Tabel penjelasan komponen antar muka utama**

| No | Nama | Jenis | Kegunaan | Tipe data |
|---|---|---|---|---|
| 1 | Alamat berkas diagram urutan | *Textfield* | Tempat untuk menampilkan lokasi berkas diagram urutan yang akan di deteksi oleh kakas bantu | String |
| 2 | Memasukkan berkas diagram urutan | *Button* | Tombol untuk memasukkan diagram urutan yang akan diperiksa ke dalam kakas bantu | - |
| 3 | Alamat Folder kode sumber | *TextField* | Tempat untuk menampilkan lokasi folder yang dipilih untuk dilakukan deteksi nantinya | String |
| 4 | Memasukkan folder kode sumber | *Button* | Tombol untuk memilih tempat folder yang berisi kode sumber | - |
| 5 | Area menampilkan hasil ekstraksi diagram urutan | *TextArea* | Tempat untuk menampilkan hasil ekstraksi dari diagram urutan yang sudah diperiksa oleh kakas bantu | String |
| 6 | Area menampilkan hasil ekstraksi kode sumber | *TextArea* | Tempt untuk menampillkan hasil ekstraksi kode sumber yang sudah diperiksa oleh kakas bantu | String |

**Tabel 3.5 Tabel penjelasan komponen antar muka utama (*lanjutan)***

| | | | | |
|---|---|---|---|---|
| 7 | Area Menampilkan hasil pendeteksian | *TextArea* | Tempat untuk menampilkan hasil dari pendeteksian yang dilakukan oleh kakas bantu | String |
| 8 | Menampilkan menu yang disediakan oleh kakas bantu | *MenuBar* | Tempat untuk menu-menu yang disediakan oleh kakas bantu. Menu itu meliputi, membuat proyek,membuka proyek, dan menutup proyek | - |

### 3.2.1.2 Jendela Masukkan Berkas Dokumen rancangan

Jendela masukkan berkas dokumen rancangan adalah antarmuka berupa sebuah jendela yang merupakan bagian dari antar muka utama kakas bantu. Komponen jendela ini, seperti daftar *folder*, *daftar file*, dan tombol "*open*". Komponen-komponen dari jendela masukkan berkas dijelaskan secara detail pada Tabel 3.6

**Tabel 3.6 Komponen jendela masukkan berkas dokumen rancangan**

| No | Nama | Jenis | Kegunaan | Tipe Data |
|---|---|---|---|---|
| 1 | Daftar folder | List | Daftar untuk menampilkan folder yang ada didalam sistem | List of folder |
| 2 | Daftar list File | List | Daftar untuk menampilkan file yang ada didalam sistem | List of file |
| 3 | Selection | Textfield | Kolom untuk menampilkan hasil dari seleksi analis | String |
| 4 | Open | Button | Mengirimkan perintah ke kakas bantu untuk melakukan seleksi file | - |

### 3.2.1.3 Jendela memilih berkas Proyek

Jendela memilih berkas proyek adalah antarmuka berupa sebuah jendela yang merupakan bagian dari antar muka utama kakas bantu. Komponen jendela ini, seperti daftar *folder*, *daftar file*, dan tombol "*open*". Komponen-komponen dari jendela masukkan berkas dijelaskan secara detail pada Tabel 3.7

**Tabel 3.7 Tabel Komponen jendela memilih berkas proyek**

| No | Nama | Jenis | Kegunaan | Tipe Data |
|---|---|---|---|---|
| 1 | Daftar folder | List | Daftar untuk menampilkan folder yang ada didalam sistem | List of folder |
| 2 | Daftar list File | List | Daftar untuk menampilkan file yang ada didalam sistem | List of file |
| 3 | Selection | Textfield | Kolom untuk menampilkan hasil dari seleksi analis | String |
| 4 | **Open** | **Button** | **Untuk mengirimkan perintah ke kakas bantu untuk melakukan seleksi file** | **-** |

### 3.2.1.4 Jendela memilih lokasi Menyimpan Proyek

Jendela memilih lokasi menyimpan proyek adalah antarmuka berupa sebuah jendela yang merupakan bagian dari antar muka utama kakas bantu. Komponen jendela, seperti daftar *folder*, *daftar file*, dan tombol "*save*". Komponen-komponen dari jendela masukkan berkas dijelaskan secara detail pada Tabel 3.8

**Tabel 3.8 Tabel Komponen Jendela memilih lokasi menyimpan Proyek**

| No | Nama | Jenis | Kegunaan | Tipe Data |
|---|---|---|---|---|
| 1 | Daftar folder | List | Daftar untuk menampilkan folder yang ada didalam sistem | List of folder |
| 2 | Selection | Textfield | Kolom untuk menampilkan hasil dari seleksi analis | String |
| 3 | Simpan | Button | Untuk mengirimkan perintah ke kakas bantu untuk melakukan penyimpanan berkas proyek pada lokasi yang sudah dipilih | - |

### 3.2.1.5 Jendela Pemilihan lokasi ekspor dokumen ".pdf"

Jendela pemilihan lokasi ekspor dokumen ".pdf" adalah antarmuka berupa sebuah jendela yang merupakan bagian dari antar muka utama kakas bantu. Komponen jendela ini, seperti daftar *folder,daftar file*, dan tombol "*save*". Komponen-komponen dari jendela masukkan berkas dijelaskan secara detail pada Tabel 3.9

**Tabel 3.9 Tabel komponen jendela pemilihan lokasi ekspor dokumen PDF**

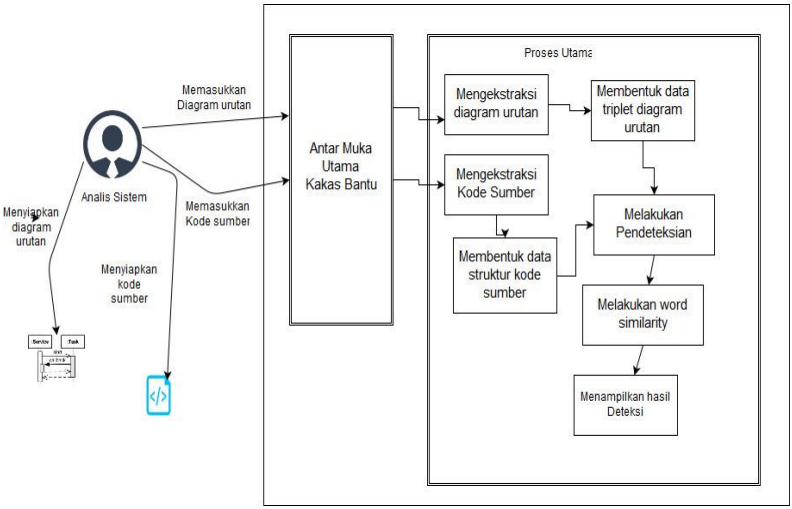
| No | Nama | Jenis | Kegunaan | Tipe Data |
|---|---|---|---|---|
| 1 | Daftar folder | List | Daftar untuk menampilkan folder yang ada didalam sistem | List of folder |
| 2 | Daftar list File | List | Daftar untuk menampilkan file yang ada didalam sistem | List of file |

**Tabel 3.9 Tabel komponen jendela pemilihan lokasi ekspor dokumen PDF (*lanjutan*)**

| | | | | |
|---|---|---|---|---|
| 3 | Selection | Textfield | Kolom untuk menampilkan hasil dari seleksi analis | String |
| 4 | Export ke pdf | Button | Untuk mengirimkan perintah ke kakas bantu untuk melakukan perubahan dari hasil kakas bantu menjadi berkas ekstensi ".pdf" | - |

### 3.2.2 Perancangan Proses Kakas Bantu

Pendeteksian ketidaksesuaian diagram urutan merupakan Proses yang terjadi di dalam kakas bantu yang hendak diimplemenntasikan.Implementasi tersebut membahas bagaimana alur dan metode untuk mendeteksi ketidaksesuaian antara kode sumber dengan diagram urutan. Secara garis besar kakas bantu ini memilikki 4 proses utama yaitu proses untuk mengekstraksi, proses pendeteksian, proses word similarity, dan proses menampilkan hasil pendeteksian. Proses utama tersebut digambarkan melalui arsitektur perangkat lunak pada Gambar 3.11




**Gambar 3.11 Gambar arsitektur kakas bantu**

Dari gambar arsitektur perangkat lunak,hubungan antara analis sistem dan kakas bantu dapat dilihat. Analis sistem memasukkan dokumen rancangan untuk di deteksi ketidaksesuaiaanya. Kemudian, kakas bantu melalui beberapa tahapan proses untuk melakukan deteksi ketidaksesuaian. Tahapan proses tersebut dijelaskan lebih detail pada sub bab berikut:

### 3.2.2.1 Menyiapkan Diagram Urutan

Diagram urutan dipersiapkan dengan memilih diagram urutan yang telah sesuai aturan, aturan yang dimaksud adalah aturan dalam pembuatan diagram urutan. Aturan tersebut dijelaskan sebagai berikut:

1. Kelas yang berinteraksi.

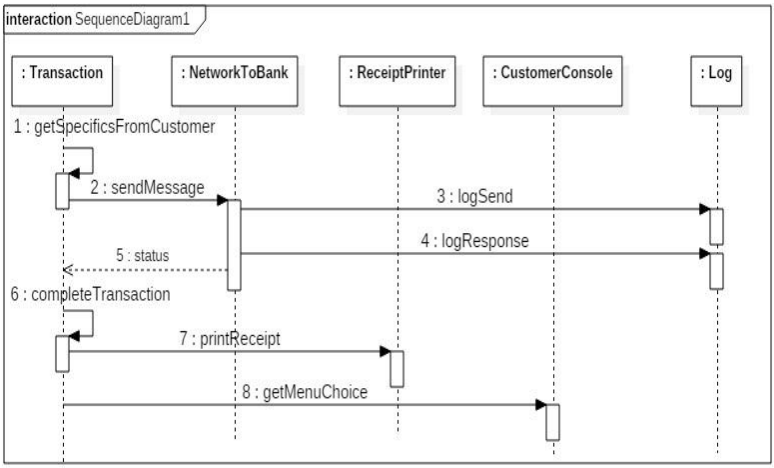
Kelas yang berinteraksi pada diagram urutan, antara lain kelas aktor, *boundary* ,*control*,dan *entity*.Adapun format dari penulisan sebuah kelas adalah titik dua ditambah dengan nama kelas dalam *upper camel-case*.

2. Pesan yang dikirim atau diterima.

Pesan berupa sebuah method yang dikirim atau diterima oleh sebuah kelas. Penulisan nama method ditulis dengan format *lower cammel-case.*Seperti contoh "matchingMethod" Setelah siap diagram urutan dipersiapkan dengan diekspor dengan menggunakan ekstensi dari aplikasi *Star-Uml*.Proses ekspor melalui menu "File -> ekspor ->XMI Export ... " pada aplikasi *Star-UML2.*

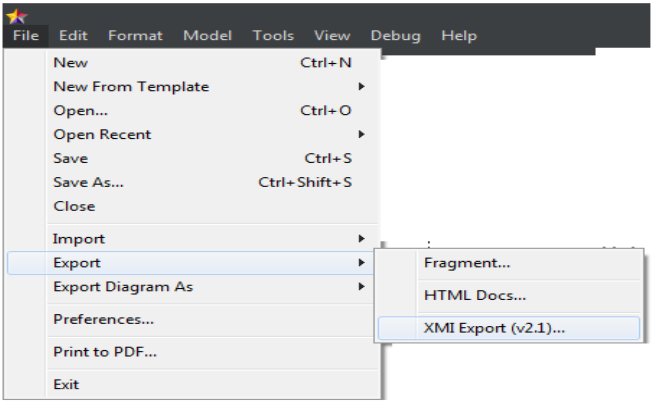
Langkah dalam mengekspor diagram urutan dijelaskan pada dibawah ini :

1. Membuka atau membuat diagram urutan di dalam Star-UML, contoh diagram urutan ditunjukkan pada Gambar 3.12




**Gambar 3.12 Contoh diagram urutan**

2. Memilih pilihan File, lalu memilih sub menu ekspor yaitu "*XMI export*". Langkah ini ditunjukkan pada Gambar 3.13




**Gambar 3.13 Langkah ekspor diagram**

3. Setelah itu pilih lokasi penyimpanan hasil ekspor diagram urutan berbentuk ".xmi"
4. Hasil ".xmi" akan berbentuk seperti ditunjukkan pada Gambar 3.14

<?xml version="1.0" encoding="UTF-8"?>

<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/spec/UML/2.0" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">

<xmi:Documentation exporter="StarUML" exporterVersion="2.0"/>

<uml:Model xmi:id="AAAAAAFgOjFj3D9uTwk=" xmi:type="uml:Model" name="RootModel">

<packagedElement xmi:id="AAAAAAFF+qBWK6M3Z8Y=" name="Model" visibility="public" xmi:type="uml:Model"/>

<packagedElement xmi:id="AAAAAAFgM+4DqTxDW5c=" name="Model1" visibility="public" xmi:type="uml:Model">

<packagedElement xmi:id="AAAAAAFgM+4YtjxI6ZI=" name="Transaction" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Class" isActive="false"/>

<packagedElement xmi:id="AAAAAAFgM+5/VDx04mk=" name="NetworkToBank" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Class" isActive="false"/>

<packagedElement xmi:id="AAAAAAFgNcut7TyhU9A=" name="ReceiptPrinter" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Class" isActive="false"/>

<packagedElement xmi:id="AAAAAAFgNeBS2T25cYU=" name="CustomerConsole" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Class" isActive="false"/>

<packagedElement xmi:id="AAAAAAFgNeB/ST3jvjQ=" name="Log" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Class" isActive="false"/>
</packagedElement>

<packagedElement xmi:id="AAAAAAFgNcwhZzzN6G4=" name="Model2" visibility="public" xmi:type="uml:Model">

……..

<packagedElement xmi:id="AAAAAAFgNcwwFTzStms=" name="Librarian" visibility="public" isAbstract="false" isFinalSpecialization="false" isLeaf="false" xmi:type="uml:Actor">

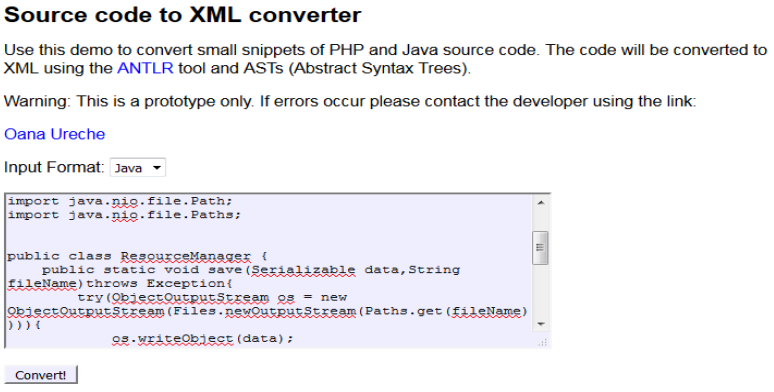
**Gambar 3.14 Contoh hasil XMI**

Setelah hasil sudah berbentuk ".xmi" maka hasil ini nanti akan bisa menjadi inputan kakas bantu ini.

### 3.2.2.2 Menyiapkan Kode Sumber

Kode sumber dipersiapkan dengan memilih kode sumber yang sesuai dibentuk berdasarkan diagram urutan. Kode sumber sendiri memilikki aturan dalam penamaan kelas dan metode. Penamaan kelas pada kode sumber menggunakan format *upper cammel-case*, contoh penggunaan "*MainClass*". Sedangkan metode menggunakan aturan lower *cammel-case*, contoh penggunaan "createMethod". Setelah siap kode sumber dipersiapkan dengan diekspor dengan menggunakan *xmltranslator* [1]. Langkah dalam penyiapan kode sumber dijelaskan pada dibawah ini:

1. Membuka halaman aplikasi *xmltranslator* pada web-browser. Halaman ini ditunjukkan pada lampiran [1]
2. Menyiapkan kode sumber java yang akan diubah ke dalam ".xml"
3. Setelah itu salin dan tempel kode sumber per berkas ke dalam halaman *xmltranslator*.
4. Pastikan input format sudah diubah menjadi java,langkah ini ditunjukkan pada gambar 3.15




**Gambar 3.15 Gambar langkah ekspor kode sumber**

5. Lalu pilih "*convert!*".Untuk setiap berkas kode sumber java harus dilakukan salin dan tempel satu persatu
6. Hasil ekspor ditunjukkan pada Gambar 3.16

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<unit language='Java'>
  <ANNOTATION_LIST>
  ANNOTATION_LIST
  </ANNOTATION_LIST>
      <IDENT>
      Receipt
      </IDENT>
    </DOT>
  </IMPORT>
  <CLASS>
  class
    <MODIFIER_LIST>
    MODIFIER_LIST
      <PUBLIC>
      public
      </PUBLIC>
      <ABSTRACT>
      abstract
      </ABSTRACT>
    </MODIFIER_LIST>
    <IDENT>
    Transaction
    </IDENT>
    <CLASS_TOP_LEVEL_SCOPE>
    CLASS_TOP_LEVEL_SCOPE
      <CONSTRUCTOR_DECL>
          <.....................>
          </.....................>
     </ CONSTRUCTOR_DECL>
</.....................>.
```

**Gambar 3.16 Hasil ekspor kode sumber**

Setelah hasil sudah berbentuk ".xmi" maka hasil ini nanti akan bisa menjadi inputan kakas bantu ini.

### 3.2.2.3 Mengekstraksi Diagram Urutan

Hasil diagram urutan yang telah diekspor sebelumnya yang ditunjukkan pada sub-bab 3.2.2.1 menjadi masukkan untuk ekstraksi diagram urutan. Ekstraksi diagram urutan mengambil attribut elemen *XML*. Berikut Tabel 3.10 menjelaskan attribut elemen *XML* yang diambil oleh ekstraksi.

**Tabel 3.10 Tabel attribut elemen XML diagram urutan**

| **Elemen XML** | **Atribut** | **Deskripsi** |
|---|---|---|
| Lifeline | Name: | Nilai atribut menunjukkan nama dari kelas |
| | xmi:id | Nilai atribut elemen ini berisikan id unik |
| | xmi:type | Nilai atribut elemen ini berisikan tipe dari tipe sebuah lifeline |
| | represents" | Nilai atribute ini berisikan sebuah id unik |
| Message | Name | Nilai atribut menunjukkan nama dari message |
| | receiveEvent | Nilai attribut menunjukkan id unik penerima message |
| | sendEvent | Nilai attribut menunjukkan id unik pengirim |
| Fragment | Xmi:id | Nilai attribut menunjukkan id unik dari fragment |
| | Covered | Nilai attribut menunjukkan id unik dari id lifeline |

**Tabel 3.10 Tabel attribut elemen XML diagram urutan (*lanjutan*)**

| | | |
|---|---|---|
| ownedAttribute | **Xmi:id** | **Nilai attribut menunjukkan id unik dari ownedAttribute** |
| | Type | Nilai attribut menunjukkan id unik dari packagedElement |
| packagedElement | Xmi:id | Nilai attribut menunjukkan id unik dari packagedElement |
| | Xmi:type | Nilai attribut menunjukkan tipe dari packagedElemet |
| | Name | Nilai attribut menunjukkan nama dari packagedElement |

Dari Tabel 3.10 nantinya ekstraksi dilakukan dengan menggunakan *SAXparser* untuk mengambil elemen-elemen *xmi*. Elemen yang diambil antara lain elemen *lifeline*, nantinya elemen ini digunakan untuk mengetahui daftar kelas yang digambarkan berupa *lifeline* dalam diagram urutan. Setelah berhasil mengambil elemen *lifeline,* dilanjutkan dengan mengambil elemen *message*, elemen ini digunakan untuk mengetahui daftar pesan yang dikirimkan ataupun diterima sebuah *lifeline*, dalam hal ini elemen *message* digambarkan sebagai metode dari sebuah kelas. Elemen berikutnya untuk diolah adalah elemen *fragment*, elemen ini menggambarkan hubungan antara elemen *lifeline* dan elemen *message* elemen ini nanti menggambarkan hubungan itu melalui *attribut* yang meliputi *xmi:id* dan *covered*,atribut *xmi:id* menggambarkan id dari sebuah *lifeline* dan *covered* menggambarkan id dari sebuah *lifeline* yang berhubungan. Elemen berikutnya untuk diolah adalah elemen *ownedAttribut* elemen *ownedAttribut*, elemen ini memberikan daftar dari atribut yang

dimilikki oleh sebuah *lifeline*, atribut *lifeline* tersebut berisi nama sebuah *lifeline*. Setelah berhasil mendapatkan elemen *ownedAttribut* kemudian dilanjutkan dengan mengesktraksi elemen *packageElement*, pada elemen ini nantinya didapatkan nama kelas yang digambarkan sebuah lifeline.

Setelah berhasil mendapatkan elemen-elemen yang didapatkan sesuai dari penjelasan diatas, elemen tersebut nantinya disimpan ke dalam sebuah *arrayList* yang selanjutnya akan diolah pada sub bab 3.2.2.5 sebagai data masukkan.

### 3.2.2.4 Mengekstraksi Kode Sumber

Hasil kode sumber yang telah dieskpor sebelumnya yang ditunjukkan pada sub-bab 3.2.2.2 menjadi masukkan untuk ekstraksi kode sumber ini. Ekstraksi kode sumber mengambil elemen *XML* yang terdapat dari berkas *XML* masing-masing kode sumber.Berikut Tabel 3.11 menjelaskan elemen XML yang diambil untuk diolah

**Tabel 3.11 Elemen xml kode sumber**

| No | Elemen | Deskripsi |
|---|---|---|
| 1 | /unit/CLASS/IDENT | Nilai dari elemen menunjukkan nama kelas |
| 2 | /unit/CLASS/MODIFIER_LIST/PUBLIC | Nilai dari elemen menunjukkan modifier dari nama kelas |
| 3 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/MODIFIER _LIST/PUBLIC | Nilai dari elemen menunjukkan modifier dari sebuah metode |

**Tabel 3.11 Elemen xml kode sumber (*lanjutan)***

| | | |
|---|---|---|
| 4 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/TYPE/QUALIFIED_TYPE_IDENT/IDENT | Nilai dari elemen menunjukkan return tipe dari sebuah metode |
| 5 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/IDENT | Nilai dari elemen menunjukkan nama dari sebuah metode |
| 6 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/BLOCK_SCOPE/VAR_DECLARATION/TYPE/QUALIFIED_TYPE_IDENT/IDENT | Nilai dari elemen menunjukkan pendeklrasian sebuah variabel inisiasi objek suatu kelas |
| 7 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/BLOCK_SCOPE/VAR_DECLARATION/VAR_DECLARATOR_LIST/VAR_DECLARATOR/IDENT | Nilai dari elemen menunjukkan nama variabel dari deklarasi objek suatu kelas |
| 8 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /FUNCTION_METHOD_DECL/BLOCK_SCOPE/EXPR/METHOD_CALL/DOT | Nilai dari elemen menunjukkan pemanggilan kode sumber yang terdapat dalam sebuah metode |
| 9 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /VOID_METHOD_DECL/MODIFIER_LIST /PUBLIC | Modifier Sebuah metode |
| 10 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /VOID_METHOD_DECL/IDENT | Nilai dari elemen menunjukkan nama dari sebuah metode |

**Tabel 3.11 Elemen xml kode sumber (*lanjutan)***

| | | |
|---|---|---|
| 11 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /VOID_METHOD_DECL/BLOCK_SCOPE/ VAR_DECLARATION/TYPE/QUALIFIED _TYPE_IDENT/IDENT | Nilai dari elemen menunjukkan pendeklarasian sebuah variabel inisiasi objek suatu kelas |
| 12 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /VOID_METHOD_DECL/BLOCK_SCOPE/ VAR_DECLARATION/VAR_DECLARAT OR_LIST/VAR_DECLARATOR/IDENT | Nilai dari elemen menunjukkan nama variabel dari deklarasi objek suatu kelas |
| 13 | /unit/CLASS/CLASS_TOP_LEVEL_SCOPE /VOID_METHOD_DECL/BLOCK_SCOPE/ EXPR/METHOD_CALL/DOT | Nilai dari elemen menunjukkan pemanggilan kode sumber dalam sebuah metode |

Dari Tabel 3.11 nantinya ekstraksi dilakukan dengan menggunakan parser *SAX* dengan metode *stack.* Nantinya setiap elemen yang telah dijelaskan pada Tabel 3.11 akan disimpan ke dalam sebuah *stack*. Dalam stack ini setiap elemen yang ditemukan dalam *xml* yang digambarkan pada Gambar 3.16 akan dimasukkan ke dalam stack, setelah selesai dimasukkan ke dalam stack ekstraksi dilanjutkan dengan menemukan elemen *xml* yang terdapat dalam Tabel 3.11 di dalam stack tersebut.

Setelah berhasil melakukan pengambilan tag pada penyimpanan *stack*, setiap elemen tag diambil isi elemen tersebut dan memasukkkan hasil elemen tersebut ke dalam *arrayList*. Selanjutnya hasil arrayList tersebut akan diolah dan dijelaskan pada bab 3.2.2.6.

### 3.2.2.5 Membentuk Data Triplet Diagram urutan

Dari sub-bab 3.2.2.3 didapatkan *arrayList* yang menyimpan elemen penjelasan dari attribut *xml*. Dari arrayList yang telah didapatkan kemudian *arrayList* dibentuk untuk membentuk sebuah triplet yang berisikan subyek, predikat, obyek.

Untuk mencari subyek didapatkan dengan cara *arrayList* yang berisi *message* diidentifikasi *sendEvent* dari *arrayList message*. Setelah didapatkan id *sendEvent* itu kemudian dicari didalam *arrayList fragment*. Didalam elemen fragmen ini nanti yang menjelaskan bagaimana sebuah kelas berinteraksi dengan message itu,didalam elemen ini *sendEvent* dicocokkan dengan *idFragment* yang berguna untuk mengambil *idFragment* yang digunakan untuk mencari nama kelas yang terdapat di dalam elemen *arrayList packageElement*. Dari hal tersebut nantinya didapatkan subyek.

Setelah berhasil mendapatkan subyek, dilanjutkan dengan mencari predikat yang digambarkan sebagai *message*. Pembentukkan message ini dimulai dengan melakukan iterasi setiap message yang ada pada sebuah *xmi* yang sudah disimpan didalam *arrayList message*. Setelah melakukan iterasi, dilakukan pengambilan *messageName* pada elemen *ArrayList message*. Dari hal tersebut nantinya didapatkan predikat

Setelah berhasil mendapatkan subyek dan predikat,dilanjutkan dengan mencari obyek yang digambarkan sebagai lifeline. pembentukan obyek ini sama dengan penjelasan pembentukan subyek. Namun yang membedakan adalah pada saat identifikasi pada elemen *arrayList message* yang diidentifikasi bukanlah *sendEvent* melainkan *receiveEvent*.

Dari berdasarkan penjelasan diatas didapatkan sebuah triplet yang telah dibentuk menjadi subyek, predikat, dan obyek. Subyek itu

sendiri merupakan kelas yang mengirim sebuah message, sedangkan predikat merupakan message, dan untuk Obyek merupakan kelas yang menerima message itu. Di dalam kasus contoh Gambar 3.12 didapatkan sebuah diagram urutan, untuk penentuan subyek, predikat, obyek telah dijelaskan sebelumnya, didapatkan sebanyak 8 buah triplet,yaitu:

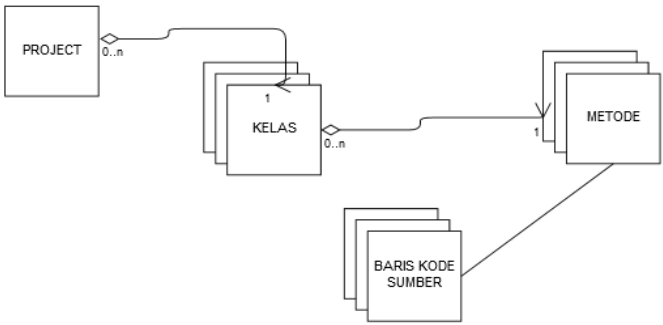
1. Transaction, getSpecificsFromCustomer, Transaction
2. Transaction, sendMessage, NetworkToBank
3. NetworkToBank, logSend, Log
4. NetworkToBank, logResponse, Log
5. NetworkToBank, status, Transaction
6. Transaction, completeTransaction, Transaction
7. Transaction, printReceipt, ReceiptPrinter
8. Transaction, getMenuChoice, CustomerConsole

Dari contoh diatas menggambarkan bagaimana pembentukkan triplet melalui diagram urutan. Untuk implementasi pembentukan triplet ini sendiri dijelaskan pada bab 4.2.2.1 yang termasuk ke dalam ekstraksi diagram urutan.

#### 3.2.2.6 Membentuk data struktur kode sumber

Dari sub-bab 3.2.2.3 didapatkan *arrayList* yang menyimpan elemen penjelasan dari attribut *xml* kode sumber. Dari elemen yang telah disimpan ke dalam *arrayList*, elemen tersebut kemudian diambil berupa nama kelas yang terdapat dalam *arrayList* elemen *project* kemudian setelah berhasil membentuk project, dilanjutkan dengan mencari metode yang terdapat dalam project tersebut. Hal itu dilakukan dengan mencari pada elemen *arrayList* metode yang diidentifikasi dengan id dari elemen *arrayList* metode sama dengan id dari elemen *arrayList project*. Setelah berhasil mendapatkan nama kelas beserta metodenya.dilanjutkan dengan pembentukkan pemanggilan kode sumber. Pembentukkan kode sumber ini

dilakukan dengan mengambil elemen dari *arrayList* baris pemanggilan kode yang telah dibentuk sebelumnya dengan mengidentifikasi id dari baris kode dengan id dari project dan metode yang ada, setelah berhasil mengindentifikasi baris kode dibentuk barulah data struktur kode sumber berhasil dibentuk. Untuk struktur dari hasil penjelasan diatas, dijelaskan pada gambar 3.17.




**Gambar 3.17 Gambar data struktur kode sumber**

Setelah berhasil melakukan pembentukan data struktur kode sumber hasil yang didapatkan dibentuk menjadi seperti pada bagian dibawah ini:

Nama kelas: public class Transaction

List metode:

1. Void getSpecificFromCustomer
2. Void completeTransaction
3. Public void performTransaction

Isi baris kode:

1. NetworkTobank.sendMessage
2. ReceiptPrinter.printReceipt

Hasil diatas merupakan contoh output dari contoh ilustrasi yang diberikan pada contoh dari *xmi* bentukan sub-bab 3.2.2.2.

### 3.2.2.7 Melakukan pendeteksian

Proses melakukan *matching* dilakukan setelah proses ekstraksi diagram urutan, proses ekstraksi kode sumber, pembentukan triplet, dan pembentukkan data struktur kode sumber yang sudah dilakukan pada sub-bab 3.2.2.5 dan 3.2.2.6. Proses ini dimulai dengan mengecek setiap hasil triplet yang sudah dihasilkan dari pembentukan 3.2.2.5 dimana setiap triplet memilikki subyek, predikat, obyek. *Matching* pertama yang dilakukan adalah melakukan *matching* subyek dari triplet terlebih dahulu. *Matching* dilakukan dengan cara mencari persamaan *character* string dari subyek dengan data kelas yang diambil dari data struktur kode sumber yang telah dibentuk pada sub-bab 3.2.2.6. Apabila ditemukan, maka dilanjutkan dengan melakukan matching apakah terdapat pemanggilan predikat dan obyek di dalam kelas subyek. Apabila terdapat maka Subyek dapat di tandai sebagai bagian triplet yang benar. Lalu setelah proses matching subyek selesai, dilanjutkan dengan matching obyek, apabila bagian triplet obyek ditemukan didalam kelas dalam data struktur, maka obyek dapat ditandai sebagai bagian triplet yang benar. Setelah itu proses matching masuk ke proses terakhir yaitu melakukan proses matching terhadap predikat. Proses dilakukan dengan mencari predikat bagian dari triplet di dalam metode yang terdapat didalam kelas obyek dari triplet. Sehingga apabila ditemukan maka triplet dapat ditandai sebagai bagian triplet yang benar. Apabila ketiga dari triplet itu benar semua maka triplet itu bisa dikatakan benar. Namun apabila salah satu dari ketiga triplet itu salah, maka triplet itu dapat dikatakan tidak sesuai dengan kode sumber yang mengartikan bahwa bagian sequence itu didapati kesalahan atau perubahan yang harus diperbarui.

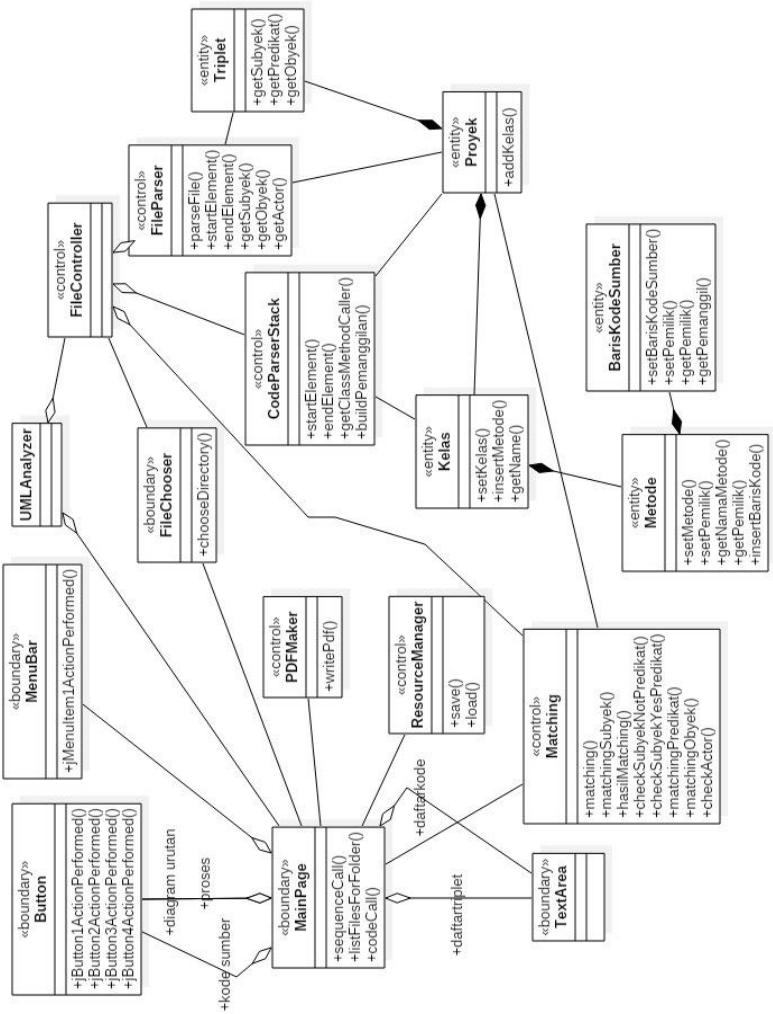
#### 3.2.2.8 Melakukan word similarity

Setelah dilakukan pendeteksian yang telah dijelaskan pada sub-bab 3.2.2.7, dilanjutkan dengan melakukan word similarity. Word similarity dimulai dengan mendapatkan hasil dari elemen arrayList yang telah didapatkan pada pembentukkan data struktur kode sumber dan elemen *arrayList* triplet diagram urutan. Setiap pendeteksian *word similarity* dilakukan untuk mendeteksi kembali predikat dengan cara mendeteksi masing-masing nama metode. Pendeteksian word similarity dilakukan pertama dengan mendeteksi predikat dengan nama metode pada kode sumber. Deteksi dilakukan pertama kali dengan melakukan *pre processing* dengan merubah suatu penamaan metode menjadi sebuah kata dan begitu sebaliknya pada predikat yang terdapat pada triplet juga. Nantinya dari kata itu dihitung jarak perbedaan makna antara kata dari predikat dan kata dari nama kelas kode sumber, apabila didapati hasil melebihi dari treshold maka didapati predikat diberi tanda benar. Untuk implementasi ini dapat ditemukan pada bab implementasi yaitu pada bab 4.

### 3.2.3 Perancangan diagram kelas

Hasil analisis kandidat kelas yang telah dilakukan sebelumnya juga digunakan untuk merancang diagram kelas kakas bantu. Diagram Kelas adalah diagram yang menggambarkan kelas-kelas yang menyusun sistem dan hubungan antar kelas-kelas tersebut. Diagram kelas disusun berdasarkan hasil model analisis setiap kasus penggunaan. Dari hasil analisis yang dilakukan ditemukan sebanyak 16 kelas utama yang menggambarkan sistem ini. Didalam kelas diagram ini menggambarkan bagaimana kelas-kelas dalam kode sumber berinteraksi dan berhubungan dalam pembuatan kakas bantu ini. Untuk penjelasan masing-masing kelas dapat dilihat pada sub-bab 3.1.9 kelas analisis. Berikut kelas

diagram yang telah dirancang sedemikian rupa ditunjukkan pada gambar 3.18




**Gambar 3.18 Gambar Perancangan kelas diagram**

# BAB IV
# IMPLEMENTASI SISTEM

Bab ini menjelaskan implementasi kakas bantu untuk mendeteksi ketidaksesuaian kode sumber dengan diagram urutan. Kakas bantu diimplementasikan dengan menggunakan bahasa pemrograman *java*.

## 4.1 Lingkungan Implementasi

Lingkungan pengembangan kakas bantu dilakukan dengan menggunakan komputer dengan spesifikasi Intel® Core™ i7-4700HQ @2.40GHz dengan memori sebesar 4GB. Selain itu, terdapat perangkat lunak yang membantu selama proses pengembangan. Perangkat lunak tersebut antara lain:

1. Sistem operasi Windows 7 64 bit
2. Kode editor Netbeans IDE 8.1 dengan *oracle java development kit*

## 4.2 Detail implementasi Kakas Bantu

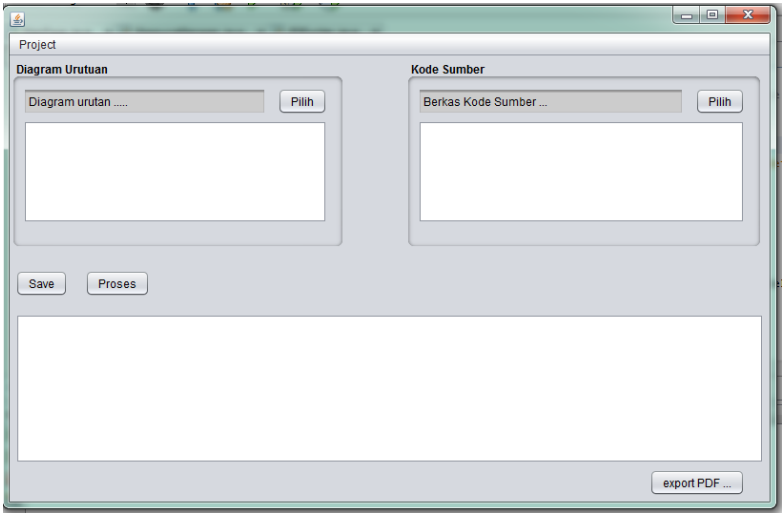
Detail implementasi kakas bantu dibagi berdasarkan perancangan antarmuka dan proses yang telah dijelaskan pada subbab perancangan sistem. Namun, tidak semua bagian implementasi tersebut dijelaskan. Implementasi antarmuka dijelaskan secara keseluruhan, sedangkan implementasi proses dijelaskan bagian yang penting saja. Bagian yang penting, misalnya implementasi "*matching*" antara triplet diagram urutan dan kode sumber, *word similarity*,implementasi ekstraksi diagram urutan, dan implementasi ekstraksi kode sumber. Penjelasan detail implementasi dari antarmuka dan proses kakas bantu dijelaskan melalui subbab berikut.

### 4.2.1 Implementasi Antarmuka

Implementasi antar muka kakas bantu dibuat dengan menggunakan pustaka *swing*. Pustaka *swing* adalah pustaka *Java graphics API* yang menyediakan kumpulan komponen *graphical user interface* (GUI) yang dapat digunakan kembali, seperti tombol, label, kolom teks, teks area, dan frame untuk membangun GUI. Implementasi antar muka menggunakan pustaka ini dengan mudah bisa dibantu oleh kode editor *NetBeans* IDE melalui GUI Builder. GUI Builder adalah sebuah *plugin* yang digunakan untuk merancang antar muka aplikasi dan melihat visualisasinya. Implementasi antar muka Kakas bantu ini meliputi:

#### 4.2.1.1 Implementasi Antar Muka Utama

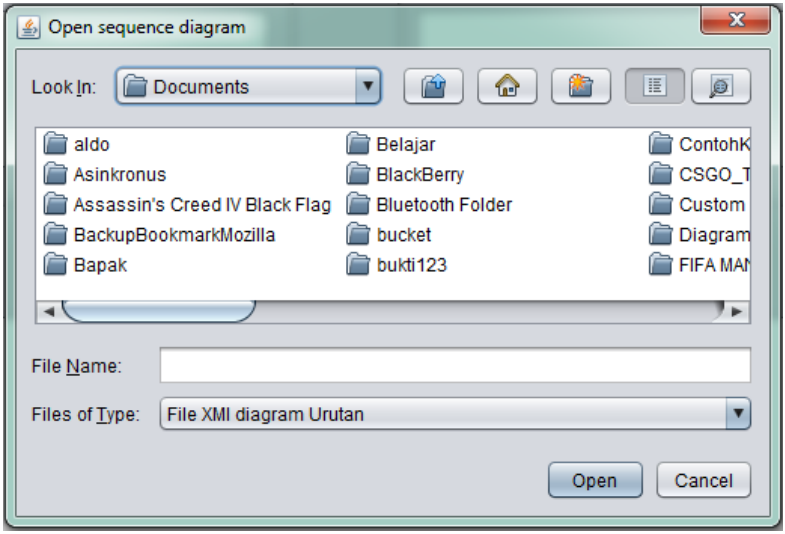
Hasil implementasi dari antarmuka utama kakas bantu ditunjukkan oleh Gambar 4.1 gambar tersebut menunjukkan jendela utama yang terdiri dari tombol, kolom teks, teks area, menubar dan komponen lain yang sesuai spesifikasi antarmuka pada Tabel 3.5. Jendela ini dalam implementasinya berupa sebuah kelas dengan nama "*MainPage*" yang mewarisi kelas "*javax.swing.Jframe*". dengan kelas tersebut komponen GUI yang disediakan oleh swing untuk membuat jendela aplikasi dapat digunakan. Antar muka utama ini nantinya akan menjadi tempat dimana proses utama dilakukan.




**Gambar 4.1 Implementasi Antar muka utama**

#### 4.2.1.2 Implementasi Jendela Masukkan Berkas Diagram Urutan dan Kode Sumber

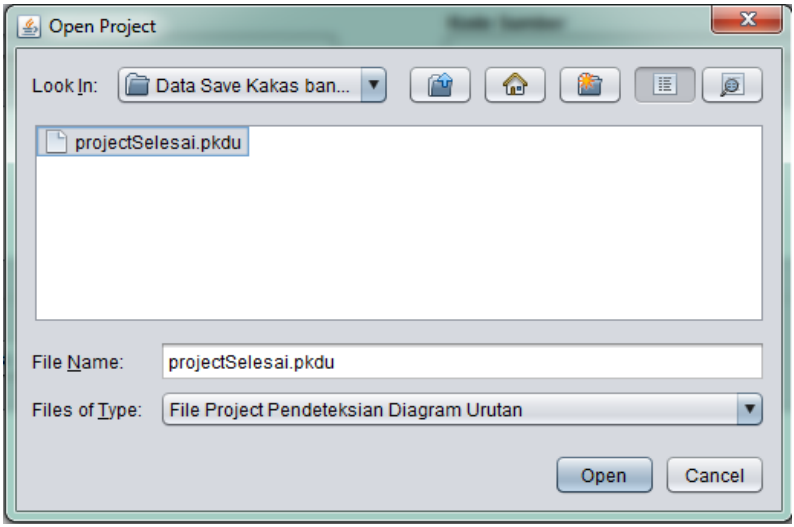
Hasil Implementasi jendela masukkan berkas dokumen rancangan dan kode sumber ditunjukkan oleh Gambar 4.2. Gambar tersebut menunjukkan komponen umum yang dipakai saat memasukkan berkas. Implementasi jendela ini sesuai spesifikasi antar muka pada Tabel 3.6. Jendela ini diimplementasikan dengan mewarisi kelas *"javax.swing.JFileChooser"*. komponen-komponen yang digunakan untuk mengatur cara kerja pemilihan berkas disediakan oleh kelas "*javax.swing.JFileChooser*". Komponen tersebut diimplementasikan, seperti pemilihan lokasi berkas dan tombol memilih berkas.




**Gambar 4.2 Implementasi Jendela Pemilihan Diagram urutan & kode sumber**

### 4.2.1.3 Implementasi Jendela memilih berkas Proyek

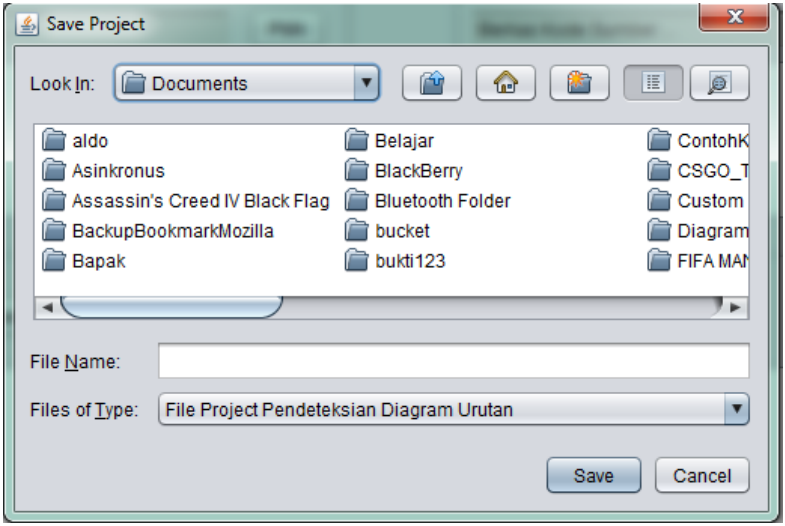
Hasil Implementasi jendela memilih berkas proyek ditunjukkan oleh Gambar 4.3. Gambar tersebut menunjukkan komponen umum yang dipakai saat memasukkan berkas proyek yang sudah disimpan. Implementasi jendela ini sesuai spesifikasi antar muka pada Tabel 3.7. Jendela ini diimplementasikan dengan mewarisi kelas "javax.swing.JFileChooser". Komponen-komponen untuk mengatur cara kerja pemilihan berkas disediakan oleh kelas "javax.swing.JFileChooser". Komponen tersebut nantinya diimplementasikan menjadi komponen pemilihan lokasi berkas dan tombol memilih berkas.




**Gambar 4.3 Implementasi Jendela Pemilihan berkas Proyek**

#### 4.2.1.4 Implementasi Jendela memilih lokasi Menyimpan Proyek

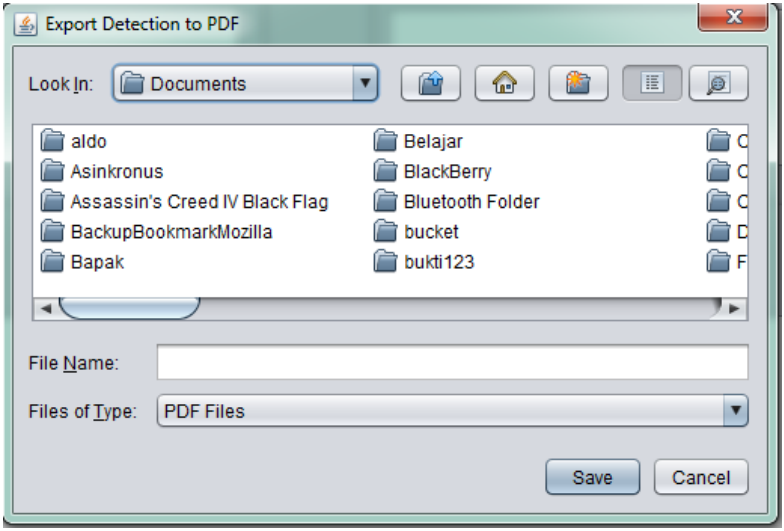
Hasil Implementasi jendela memilih lokasi menyimpan proyek ditunjukkan oleh Gambar 4.4. Gambar tersebut menunjukkan komponen umum yang dipakai saat memasukkan berkas proyek yang sudah disimpan. Implementasi jendela ini sesuai spesifikasi antar muka pada Tabel 3.8. Jendela ini diimplementasikan dengan mewarisi kelas "javax.swing.JFileChooser". Komponen-komponen untuk mengatur cara kerja pemilihan berkas disediakan oleh kelas "javax.swing.JFileChooser". Komponen tersebut diimplementasikan, seperti pemilihan lokasi berkas dan tombol memilih berkas.




**Gambar 4.4 Implementasi Jendela memilih lokasi menyimpan proyek**

#### 4.2.1.5 Implementasi Jendela Pemilihan lokasi penyimpanan ekspor dokumen ".pdf"

Hasil Implementasi jendela memilih lokasi penyimpanan ekspor dokumen ditunjukkan oleh Gambar 4.5. Gambar tersebut menunjukkan komponen umum yang dipakai saat memasukkan berkas proyek yang sudah disimpan.Implementasi jendela ini sesuai spesifikasi antar muka pada Tabel 3.9. Jendela ini diimplementasikan dengan cara mewarisi kelas-kelas yang dimilikki oleh pustaka *"javax.swing.JFileChooser"*. Komponen-komponen untuk mengatur cara kerja pemilihan berkas disediakan oleh kelas-kelas yang dimilikki oleh pustaka *"javax.swing.JFileChooser"*. Komponen tersebut nantinya diimplementasikan menjadi komponen pemilihan lokasi berkas dan tombol memilih berkas.




**Gambar 4.5 Implementasi Jendela memilih penyimpanan ekspor dokumen ".pdf"**

### 4.2.2 Implementasi Proses

Sesuai dengan penjelasan sebelumnya,detail implementasi proses yang ada di dalam kakas bantu tidak dijelaskan secara keseluruhan. Bagian implementasi proses yang penting saja dijelaskan. Implementasi proses tersebut antara lain:

#### 4.2.2.1 Implementasi Ekstraksi Diagram urutan

Berkas ".xmi" diagram urutan yang telah dimasukkan ke dalam kakas bantu diekstraksi isinya saat analis sistem memasukkan diagram urutan ke kakas bantu. Proses ekstraksi diagram urutan ditunjukkan oleh fungsi *parseFile*. Fungsi *parseFile* ditunjukkan pada potongan kode berikut ini

```
01.  public void parseFile(String filename) throws SAXException,IOException {
02.                          //Finally, tell the parser to parse the input and notify the handler
03.          sp.parse(filename, this);
04.      }
```

***Kode Sumber 4.1 Potongan kode fungsi parseFile***

Untuk bisa menjalankan potongan fungsi kode diatas, diperlukan sebuah instansi baru dari objek *SAXParserFactory* . instansi objek itu dapat dilihat dari potongan kode dibawah ini.

```
01.  public FileParser() throws IOException, SAXException,ParserConfigurationException, org.xml.sax.SAXException  {
02.             SAXParserFactory spfac = SAXParserFactory.newInstance();
03.              sp = spfac.newSAXParser();
04.    }
```

***Kode Sumber 4.2  Potongan kode fungsi instansi objek SAXParserFactory***

Fungsi *parseFile* kemudian melakukan ekstraksi, melalui instansi objek diatas. ekstraksi diagram urutan ini dilakukan dalam format ".xmi" dibantu oleh SAX (*the simple* API *for* XML). SAX merupakan salah satu pustaka untuk memparsing dokumen XML. SAX ini mengambil elemen dokumen XML dengan membaca berkas dari atas kebawah.Proses ekstraksi elemen dokumen ditunjukkan oleh fungsi *parseFile* pada baris ke 2 pada potongan kode sumber 4.1, fungsi itu mengimplementasi API (*Application Programming Interface*) yang disediakan oleh pustaka SAX. Implementasi itu ditunjukkan pada potongan kode dibawah ini. Dalam fungsi kode sumber 4.1 memanggil fungsi parse dimana fungsi parse itu akan menjalankan proses pembacaan dari atas ke bawah,pembacaan dari atas kebawah itu dilakukan melalui fungsi start elemen dan end elemen yang ditunjukkan pada kode sumber 4.3 dan 4.4.Fungsi 4.3 dan 4.4 merupakan hasil dari *instance* yang ditunjukkan pada potongan kode sumber 4.2. Kode sumber 4.2 menunjukkan *instance object* SAX yang berguna untuk mengimplementasi metode parse pada kode sumber 4.1 yang akan digunakan pada pembuatan kakas bantu ini.

```
1.  public void startElement(String uri, String localName,String qName, Attributes attr
    ibutes) throws SAXException {
2.              temp = "";
3.              if (qName.equalsIgnoreCase("lifeline")) {
4.                      acct = new Account();
5.                      String packageName = attributes.getValue("name");
6.                      String packageType = attributes.getValue("xmi:type");
7.                      String PackageID = attributes.getValue("xmi:id");
8.                      String representId = attributes.getValue("represents");
9.                      acct.setRepresent(representId);
10.                     acct.setType(packageType);
11.                     acct.setName(packageName);
12.                     acct.setId(PackageID);
13.
14.             }
15.             else if (qName.equalsIgnoreCase("message")) {
16.                     msgtriplet = new MessageTriplet();
17.                     String messageName = attributes.getValue("name");
18.                     String receiveEvent = attributes.getValue("receiveEvent");
19.                     String sendEvent = attributes.getValue("sendEvent");
20.                     msgtriplet.setMethod(messageName);
21.                     msgtriplet.setIdMessage(receiveEvent);
22.                     msgtriplet.setCoveredId(sendEvent);
23.
24.             }
25.              else if (qName.equalsIgnoreCase("fragment")) {
26.                   fgmm = new fragment();
27.                   String messageEvent = attributes.getValue("xmi:id");
28.                   String lifelineCovered = attributes.getValue("covered");
29.                   fgmm.setId(messageEvent);
30.                   fgmm.setCovered(lifelineCovered);
31.
32.             }
33.              else if(qName.equalsIgnoreCase("ownedAttribute")){
34.                  ownedAttr = new OwnedAttributes();
35.                  String xmIdOwned = attributes.getValue("xmi:id");
36.                  String idPackageOwned = attributes.getValue("type");
37.                  ownedAttr.setOwnedAttributes(xmIdOwned, idPackageOwned);
38.              }
39.              else if(qName.equalsIgnoreCase("packagedElement")){
40.                  //untuk mengambil daftar actor lalu masukkan ke fungsi get actor
41.                  packageElement = new PackagedElement();
42.                  String idFragment = attributes.getValue("xmi:id");
43.                  String tipeUML = attributes.getValue("xmi:type");
44.                  String namaPackage = attributes.getValue("name");
45.                  packageElement.setPackagedElement(tipeUML,idFragment, namaPackage);

46.              }
47.
48.     }
```

***Kode Sumber 4.3 fungsi start elemen***

```
1.
2.   public void endElement(String uri, String localName, String qName)throws SAXExcept
     ion {
3.
4.          if (qName.equalsIgnoreCase("lifeline")) {
5.                  // add it to the list
6.                  accList.add(acct);
7.            }
8.          else if (qName.equalsIgnoreCase("message")) {
9.                  // add it to the list
10.                 listmsgtriplet.add(msgtriplet);
11.                 msg.add(msgg);
12.
13.           }
14.          else if (qName.equalsIgnoreCase("fragment")) {
15.                 // add it to the list
16.                 fgm.add(fgmm);
17.           }
18.          else if(qName.equalsIgnoreCase("ownedAttribute")){
19.              owdElement.add(ownedAttr);
20.          }
21.          else if(qName.equalsIgnoreCase("packagedElement")){
22.              pkgElement.add(packageElement);
23.          }
24.
25.
26.    }
```

***Kode Sumber 4.4 fungsi end elemen***

Fungsi *startElement* pada baris ke 1 pada kode sumber 4.3 memberikan tanda untuk memulai ekstraksi elemen yang ada pada berkas ".xmi" elemen tersebut dimulai saat pembacaan *startElement* melalui tag pembuka pada xml.Lalu pada beris ke 3 memulai ekstraksi elemen *lifeline* pada XMI diagram urutan.Informasi yang diambil dari elemen ini adalah atribut elemen seperti *nama*, *xmi:type,xmi:id*, dan *represents*.Setiap informasi itu nantinya disimpan ke dalam *arrayList* dengan data

model *acct* yang ada pada baris ke 4.Setelah itu ekstraksi dilanjutkan dengan mengambil elemen yang telah dijabarkan pada bab 3.2.2.3.Fungsi *endElement* pada baris ke 2 kode sumber 4.4 berfungsi sebaliknya sebagai penanda berhentinya ekstraksi elemen diagram urutan.Fungsi endElement ini berjalan ketika ditemukan tag penutup elemen pada ".xmi".atribut elemen yang sebelumnya sudah disimpan menjadi kelas model *acct* kemudian dimasukkan ke dalam arraylist yang ditunjukkan pada baris 6.Hal yang sama juga berlaku untuk elemen-elemen yang dijelaskan pada sub-bab 3.2.2.3 dan Tabel 3.22

Setelah berhasil mendapatkan elemen-elemen diatas dan berhasil disimpan ke dalam *arrayList* Proses dilanjutkan dengan pembentukkan hasil elemen diagram urutan menjadi sebuah triplet.Fungsi pembentukan sebuah triplet digambarkan pada fungsi kode sumber 4.5.

```
01. public void creatinTriplet(){
02.     String Subyek;
03.     String predikat;
04.     String Obyek;
05.     int size = listmsgtriplet.size();
06.     System.out.println(size);
07.     for(int i= 0;i<size;i++)
08.     {
09.         String isiObyek = listmsgtriplet.get(i).getIdMessage().toString();
10.          triplet tpl = new triplet();
11.          tripletku.add(tpl);
12.          predikat = listmsgtriplet.get(i).getMethod().toString();
13.          Subyek = getSubyek(listmsgtriplet.get(i).getCoveredId().toString());
14.          Obyek = getObyek(isiObyek);
15.
16.          Triplet tripletData = new Triplet(Subyek,Obyek,predikat);
17.          sequenceData.insertTriplet(tripletData);
18.          tpl.setObyek(Obyek);
19.          tpl.setPredikat(predikat);
20.          tpl.setSubyek(Subyek);
21.          tripletku.add(tpl);
22.    }
23. }
```

***Kode Sumber 4.5 potongan kode pembuatan triplet***

Dari potongan kode sumber 4.5 dibentuk sebuah triplet dimana triplet dibentuk dengan cara setiap iterasi elemen *arrayList message* yang ditemukan maka diambil *idMessage* yang menggambarkan id *receiveEvent* yang merupakan kelas penerima dari sebuah *message*.Selanjutnya *getMethod* yang menggambarkan *message* dikirimkan atau diterima oleh sebuah kelas.Kemudian *getCoveredId* yang menggambarkan id *senderEvent* yang merupakan kelas pengirim dari sebuah message.Nantinya dari ketiga hal ini akan terbentuk sebuah triplet,Namun pada fase ini triplet terbentuk dengan sebuah id yang menggambarkan hal yang telah dijelaskan diatas tadi.hasil sementara untuk potongan kode 4.5 ditunjukkan pada gambar 4.6

```
AAAAAAFgOjFj3T9viu4= getSpecificsFromCustomer AAAAAAFgOjFj3T9wvDc=
AAAAAAFgOjFj3T9xuJM= sendMessage AAAAAAFgOjFj3T9y6S0=
AAAAAAFgOjFj3T9Oz6c= logSend AAAAAAFgOjFj3T9iAaY=
AAAAAAFgOjFj3T935ME= logResponse AAAAAAFgOjFj3T94c6Q=
AAAAAAFgOjFj3j96+ao= status AAAAAAFgOjFj3j975rQ=
AAAAAAFgOjFj3j98m9A= completeTransaction AAAAAAFgOjFj3j99+iM=
AAAAAAFgOjFj3j9+p7Q= printReceipt AAAAAAFgOjFj3j9/vPI=
AAAAAAFgOjFj3j+A5HQ= getMenuChoice AAAAAAFgOjFj3j+B8KI=
```

**Gambar 4.6 Gambar hasil sementara triplet**

Dari hasil pada gambar 4.6 nantinya diperlukan sebuah metode ntuk mengetahui nama kelas dari id terebut,maka diperlukan sebuah fungsi untuk mencari id tersebut dari elemen arrayList fragment,lalu setelah ditemukan id dari elemen arrayList fragment dilanjutkan mencari id dari elemen *arrayList ownedAttribut* dari id *ownedAttribut* ini nantinya dapat mengetahui isi dari nama id tersebut dari *packageEelemen* yang telah disimpan dalam *arrayList* sebelumnya.Dalam hal ini penjelasan digambarkan ke kode sumber pada kode sumber 4.6 dan 4.7.

```
1.  private String getObyek(String obyek) {
2.           .....
3.          for(int x=0;x<sizeMessageTriplet;x++)
4.          {
5.             for(int y=0;y<sizefragment;y++){
6.               ......
7.                  for(int i=0;i<size;i++){
8.                         ....
9.                        if(obyek.equals(listFragmentTemp.get(y).getIdFragment().toStr
    ing())){
10.                              if(listFragmentTemp.get(y).getIdCovered().toString().eq
    uals(listLifelineTemp.get(i).getId().toString())){
11.                               String idRepresentLifeline = accList.get(i).getRepres
    ent().toString();
12.                                  for(int j=0;j<owdElement.size();j++){
13.                                     String idOwned = owdElement.get(j).getIdOwnedAtt
    ributes();
14.                                     if(idRepresentLifeline.equals(idOwned)){
15.                                     String idTypeOwdElement = owdElement.get(j).getI
    dTypePackageElement();
16.                                          for(int h=0;h<pkgElement.size();h++){
17.                                              String idPackagedElement = pkgElement.g
    et(h).getId();
18.                                              if(idTypeOwdElement.equals(idPackagedEl
    ement)){
19.                                                   obyek = pkgElement.get(h).getName(
    );
20.                                              }
21.                                          }
22.                                     }
23.                                  }
24.
25.                              }
26.                          }
27.                 }
28.            }
29.          }
30.          return obyek;
31.      }
```

**Kode Sumber 4.6 Fungsi getObyek**

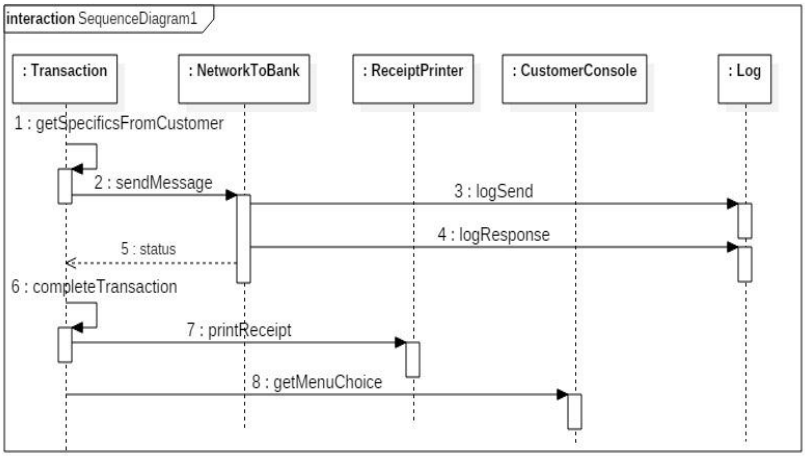
```
1.  private  String getSubyek(String subyek) {
2.          int size = accList.size();
3.          int sizefragment = fgm.size();
4.          String idFragment;
5.          String coveredId;
6.          int sizeMessageTriplet = listmsgtriplet.size();
7.          for(int y=0;y<sizeMessageTriplet;y++){
8.            for(int x=0;x<sizefragment;x++){
9.             idFragment = fgm.get(x).getId().toString();
10.            coveredId = fgm.get(x).getCovered().toString();
11.            tempFragment.setIdFragment(idFragment);
12.            tempFragment.setIdCovered(coveredId);
13.            listFragmentTemp.add(tempFragment);
14.             for(int i=0;i<size;i++){
15.              String idLifeline = accList.get(i).getId().toString();
16.              String idRepresentLifeline = accList.get(i).getRepresent().toString();

17.              tempLifeline.setId(idLifeline);
18.              tempLifeline.setRepresent(idRepresentLifeline);
19.              listLifelineTemp.add(tempLifeline);
20.              if(subyek.equals(listFragmentTemp.get(x).getIdFragment().toString()))

21.              {                 if(listFragmentTemp.get(x).getIdCovered().toString().
    equals(listLifelineTemp.get(i).getId().toString()))
22.                     {
23.                       for(int j=0;j<owdElement.size();j++){
24.                             String idOwnedAttribute = owdElement.get(j).getIdOwnedAt
    tributes();
25.                              if(idRepresentLifeline.equals(idOwnedAttribute))
26.                                  {
27.                                    String idTypeOwdElement = owdElement.get(j).getI
    dTypePackageElement();
28.                                          for(int h=0;h<pkgElement.size();h++){
29.                                          String idPkgElement = pkgElement.get(h).ge
    tId();
30.                                          if(idTypeOwdElement.equals(idPkgElement)){

31.                                                      subyek = pkgElement.get(h).get
    Name();
32.                                                 }
33.                                             }
34.                                          }
35.                                       }
36.
37.                                     }
38.                               }
39.
40.                     }
41.             }
42.         }
43.         return subyek;
44.     }
```

**Kode Sumber 4.7 Kode fungsi getSubyek**

Dari hasil penjelasan,didapatkan pembentukkan diagram urutan menjadi sebuah triplet.Triplet tersebut nantinya akan menggambarkan alur dari diagram urutan,yang akan dijadikan pembanding kode sumber dalam mencari ketidaksesuaian kode sumber.Hasil dari proses ekstraksi ini digambarkan menjadi sebuah triplet seperti dibawah ini:

1. Transaction, getSpecificsFromCustomer, Transaction
2. Transaction, sendMessage, NetworkToBank
3. NetworkToBank, logSend, Log
4. NetworkToBank, logResponse, Log
5. NetworkToBank, status, Transaction
6. Transaction, completeTransaction, Transaction
7. Transaction, printReceipt, ReceiptPrinter
8. Transaction, getMenuChoice, CustomerConsole

Hasil diatas didapatkan dari diagram urutan yang digambarkan pada gambar 4.6




**Gambar 4.7 Diagram urutan sebelum ekstraksi**

#### 4.2.2.2 Implementasi Ekstraksi Kode Sumber

Setelah melakukan ekstraksi diagram.Berkas kode sumber yang telah dimasukkan ke dalam kakas bantu diekstraksi isinya saat analis sistem memasukkan lokasi kode sumber ke kakas bantu. Proses ekstraksi kakas bantu ditunjukkan oleh fungsi *codeCall*. fungsi *codeCall*. Ditunjukkan pada potongan kode berikut ini.

```
01.  public void codeCall(String xmlfile){
02.          try {
03.             XMLReader xr = XMLReaderFactory.createXMLReader();
04.             // Set the ContentHandler...
05.             xr.setContentHandler(cps);
06.             // Parse the file...
07.              xr.parse( new InputSource(new FileReader(xmlfile)));
08.          } catch(Exception e) {
09.              e.printStackTrace();
10.          }
11.      ......
```

***Kode Sumber 4.8 potongan kode fungsi codeCall***

Fungsi *codeCall* memilikki parameter *xmlfile* yang merupakan alamat lokasi dimana berkas kode sumber disimpan.Ekstraksi kode sumber dilakukan dengan format ".xml" dibantu dengan XMLReader.XMLReader sendiri merupakan sebuah pustaka untuk memparsing dokumen XML.Pustaka ini bekerja hampir sama dengan SAX, pustaka ini membaca dari atas ke bawah.Proses ekstraksi ditunjukkan oleh fungsi *xr.parse* pada baris kode ke 7. Fungsi ini mengimplementasi API(*Application programming interface*) yang disediakan oleh XMLReader.Implementasi tersebut ditunjukkan oleh kode sumber 4.5

```
01.  public void startElement( String namespaceURI,
02.                String localName,
03.               String qName,
04.               Attributes attr ) throws SAXException {
05.        contents.reset();
06.       tagStack.push( localName );
07.
08.
09.    }
10.    public void endElement( String namespaceURI,
11.                String localName,
12.               String qName ) throws SAXException {
13.
14.
15.       if ( getTagPath().equals( "/unit/CLASS/IDENT" ) ) {
16.          isiNamaClass = contents.toString().trim();
17.          String tipeClass = "class";
18.          kelas.setKelas(isiNamaClass, isiModifierClass, tipeClass);
19.          proyek.addKelas(kelas);
20.       }
21.
22.       if(getTagPath().equals("/unit/CLASS/MODIFIER_LIST/PUBLIC")){
23.           isiModifierClass = contents.toString().trim();
24.       }
25.       //FUNCTION THAT HAS RETURN VALUE
26.       if(getTagPath().equals("/unit/CLASS/CLASS_TOP_LEVEL_SCOPE/FUNCTION_METHOD_DECL/MODIFIER_LIST/PUBLIC")){
27.           methodPublicTemp = contents.toString().trim();
28.       }
29.       else if(getTagPath().equals("/unit/CLASS/CLASS_TOP_LEVEL_SCOPE/FUNCTION_METHOD_DECL/TYPE/QUALIFIED_TYPE_IDENT/IDENT")){
30.           typePublicTemp = contents.toString().trim();
31.       }
32.       else if(......){
33.           ........
34.       }
35.
36.     }
```

***Kode Sumber 4.9 Potongan kode fungsi startElement dan endElement***

Pada ekstraksi kode sumber ini,ekstraksi dilakukan dengan metode pendekatan yang berbeda terhadap ekstraksi diagram urutan.dalam metode ekstraksi kode sumber ini,ekstraksi dilakukan dengan menggunakan metode *stack* dimana elemen-elemen XML yang ditemukan disimpan ke dalam stack terlebih dahulu,penyimpanan elemen ke dalam *stack* dapat ditemukan pada baris kode ke 6. Kemudian ekstraksi masing-masing elemen dilakukan pada fungsi *endElement*,didalam fungsi *endElement* ekstraksi dilakukan dengan mencari elemen yang sudah dimasukkan ke dalam stack dengan fungsi *getTagPath*,potongan kode *getTagPath* dapat dilihat dari potongan kode 4.10

```
01.  private String getTagPath( ){
02.        String buffer = "";
03.        Enumeration e = tagStack.elements();
04.        while( e.hasMoreElements()){
05.                  buffer  = buffer + "/" + (String) e.nextElement();
06.
07.        }
08.        return buffer;
09.     }
```

***Kode Sumber 4.10 Potongan kode fungsi getTagPath***

Dari fungsi *getTagPath* ini mengeluarkan elemen-elemen yang sudah dimasukkan ke dalam stack,hal ini ditunjukkan pada baris kode ke 5 pada kode sumber 4.10.Setelah itu setiap elemen kode sumber yang ditemukan nilainya disimpan ke dalam variable,yang ditunjukkan oleh Kode Sumber 4.9 pada baris kode 16. Setelah itu nilai yang disimpan akan dimasukkan ke dalam sebuah kelas model yang ditunjukkan pada baris kode 18 pada kode sumber 4.9.Setelah disimpan ke dalam kelas model,selanjutnya setelah elemen disimpan ke dalam kelas model,elemen itu dimasukkan ke dalam sebuah *arrayList* yang dibentuk menjadi sebuah data struktur yang ditunjukkan pada baris kode ke 18 dan 19.Hal yang sama dilakukan untuk elemen-elemen yang sudah dijelaskan sebelumnya pada sub-bab 3.2.2.4.

Setelah berhasil memasukkan hasil kelas mode ke dalam sebuah *arrayList* dilanjutkan dengan pembentukan struktur data pemanggilan kode sumber yang digambarkan pada kode sumber 4.11

```
1.  public void getClassMethodCaller(){
2.          ......
3.           for(int x=0;x<sizeInisiasiCode;x++){
4.                  .......
5.                for(int y=0;y<sizeInitCode;y++){
6.                       ........
7.                   if(namaClassTemp.equalsIgnoreCase(namaClassInisiator)){
8.                        .....
9.                        barisKodeData = new BarisKodeSumber();
10.                       barisKodeData.setBarisKodeSumber(simpanNamaClass, simpanNamaMeto
    deClass);
11.                       barisKodeData.setPemilik(namaPemilik);
12.                       metodeData.insertBarisKode(barisKodeData);
13.                  }
14.              }
15.          }
16.     }
```

**Kode Sumber 4.11 Fungsi pembentukkan pemanggilan kode sumber**

Dari kode sumber 4.11 didapatkan pembentukan pemanggilan kode sumber dibentuk dari elemen *arrayList inisiasiCode* dan elemen *arrayList initCode* yang merupakan arrayList yang menampung daftar nama kelas yang telah disimpan pada fungsi kode sumber 4.9.Setelah itu elemen diiterasi mencari nama kelas yang sama,hal ini ditunjukkan pada baris 7 pada kode sumber 4.11.Setelah berhasil ditemukan iterasi tersebut di masukkan ke dalam kelas model *barisKodeData* dan dimasukkan ke *arrayList metodeData*.Setelah iterasi selesai maka baris pemanggilan kode sumber berhasil dibentuk,Begitu juga nama kelas,nama metode sehingga struktur data kode sumber berhasil dibentuk.

Hasil dari proses ekstraksi ini digambarkan menjadi sebuah struktur data dimana terdapat nama kelas,nama metode didalam kelas,dan pemanggilan metode dalam kelas lain.Hasil tersebut ditunjukkan pada gambar 4.7

Nama Kelas : public class NetworkToBank
List Metode :
1. public void openConnection
2. public void closeConnection
3. public void sendMessage
Isi Baris Kode :
1.Log.logSend didalam sendMessage

Nama Kelas : public class OperatorPanel
List Metode :
1. public Money getInitialCash
Nama Kelas : public class ReceiptPrinter
List Metode :
1. public void printReceipt

Nama Kelas : public class Transaction
List Metode :
1. void getSpecificsFromCustomer
2. void completeTransaction
3. public null getSerialNumber
Isi Baris Kode :
1.NetworkToBank.sendMessage didalam getSpecificsFromCustomer
2.ReceiptPrinter.printReceipt didalam completeTransaction
3.NetworkToBank.sendMessage didalam performTransaction

**Gambar 4.8 Hasil ekstraksi kode sumber**

### 4.2.2.3 Implementasi Pendeteksian

Implementasi pendeteksian dilakukan setelah analis sistem sudah melakukan ekstraksi diagram urutan dan kode sumber.Setelah itu proses pendeteksian dilakukan.Proses pendeteksian dilakukan berdasarkan penjelasan dari pendeteksian matching pada bab 3.2.2.7 yang menjelaskan bagaimana pendeteksian matching dilakukan.Proses pendeteksian ditunjukkan oleh fungsi *hasilMatching* yang ditunjukkan pada kode sumber 4.12

```
01.  public void hasilMatching(){
02.          for(int x=0;x<isiTriplet.size();x++){
03.              checkActor(isiTriplet.get(x).getSubyek());
04.              if(setActor == true){
05.                  subyek = isiTriplet.get(x).getSubyek() + " (V) ";
06.                  predikat = matchingPredikat(isiTriplet.get(x).getPredikat());
07.                  checkActor(isiTriplet.get(x).getPredikat());
08.                   if(setActor == true){
09.                       predikat = isiTriplet.get(x).getPredikat()+" (X) ";
10.                       obyek = isiTriplet.get(x).getObyek()+" (X) ";
11.                   }
12.                   else{
13.                        predikat = matchingPredikat(isiTriplet.get(x).getPredikat());
14.                        obyek = matchingObyek(isiTriplet.get(x).getObyek(),isiTriplet.get(x).getPredikat());
15.                   }
16.              }
17.              else {
18.              subyek = matchingSubyek(isiTriplet.get(x).getSubyek().toString(),isiTriplet.get(x).getPredikat(),isiTriplet.get(x).getObyek());
19.              checkActor(isiTriplet.get(x).getPredikat());
20.                  if(setActor == true){
21.                  predikat = isiTriplet.get(x).getPredikat()+" (X) ";
22.                  obyek = isiTriplet.get(x).getObyek()+" (X) ";
23.                  }
24.                  else {
25.                predikat = matchingPredikat(isiTriplet.get(x).getPredikat());
26.                obyek = matchingObyek(isiTriplet.get(x).getObyek(),isiTriplet.get(x).getPredikat());
27.                  }
28.              }
29.              String hasilAkhir = subyek+" "+obyek+" "+predikat;
30.              hasilMatchingnya.add(hasilAkhir);
31.              System.out.println(subyek+" "+obyek+" "+predikat);
32.      }
33.     }
```

***Kode Sumber 4.12 Potongan kode Fungsi pendeteksian***

Dari potongan kode diatas, dilakukan pendeteksian dengan melakukan string matching setiap isi dari triplet yang ada pada isi list *isiTriplet* yang ditunjukkan pada baris kode ke 2 pada kode sumber 4.12.Pendeteksian dilakukan dengan mendeteksi apakah subyek sebuah aktor atau tidak,hal itu ditunjukkan pada baris kode ke 3.Apabila subyek merupakan aktor maka akan dilanjutkan pada baris kode ke 4.Sedangkan apabila aktor bukan dari subyek maka dilanjutkan pada baris kode ke 17.Setelah selesai mendeteksi aktor atau bukan kemudian dilanjutkan dengan pendeteksian predikat dan obyek.Untuk pendeteksian predikat dan obyek ditunjukkan pada baris kode 13 dan baris kode 14.Untuk pendeteksian subyek apabila subyek bukan aktor ditunjukkan pada baris kode ke 18. Untuk masing-masing obyek,dan predikat dapat dilihat pada potongan kode 4.13, 4.14.

```
01.  private String matchingPredikat(String predikat) {
02.        String predikatMantap=null;
03.         for(int a=0;a<isiKelasBaru.size();a++){
04.                 String namaKelasCekObyek = isiKelasBaru.get(a).getName();
05.                     if(predikat.equals(namaKelasCekObyek)){
06.                        predikatMantap = predikat+" (V) ";
07.                        break;
08.                     }
09.                     else{
10.                         predikatMantap = predikat+" (X) ";
11.                     }
12.         }
13.      return predikatMantap;
14.     }
```

***Kode Sumber 4.13 Potongan kode fungsi deteksi Predikat***

```
01.  private String matchingObyek(String obyek3, String predikat1) {
02.        String obyekMantap=null;
03.         for(int x=0;x<isiMetode.size();x++){
04.                 String metodenyaPemilikPredikat = isiMetode.get(x).getPemilik();
05.                 String metodenyaPredikat = isiMetode.get(x).getNamaMetode();
06.                   if(predikat1.equals(metodenyaPemilikPredikat)){
07.                       if(obyek3.equalsIgnoreCase(metodenyaPredikat)){
08.                           obyekMantap = obyek3+" (V) ";
09.                           break;
10.                     }
11.                       else{
12.                           obyekMantap = obyek3+" (X) ";
13.                       }
14.                  }
15.                   else{
16.                       obyekMantap = obyek3+" (X) ";
17.                   }
18.              }
19.         return obyekMantap;
20.     }
```

***Kode Sumber 4.14 Potongan kode fungsi deteksi Obyek***

Untuk deteksi subyek ditunjukkan pada Kode sumber 4.15

```
01.  private String matchingSubyek(String subyek, String predikat1, String obyek1) {
02.   String subyekMantap = null;
03.   if(predikatSebelum==null){
04.       subyekMantap = checkSubyekNotPredikat(subyek, predikat1, obyek1);
05.   }
06.   else{
07.       if(subyek.equalsIgnoreCase(predikatSebelum)){
08.           if(!isiMetode.isEmpty()){
09.               subyekMantap = checkSubyekYesPredikat(subyek, predikat1, obyek1);
10.           }}
11.
12.       else{
13.        subyekMantap = checkSubyekNotPredikat(subyek, predikat1, obyek1);
14.        }
15.   }
16.      return subyekMantap;
17.   }
```

***Kode Sumber 4.15 Potongan kode fungsi deteksi Subyek***

Pada deteksi subyek ini dilakukan dengan mendeteksi apakah terdapat obyek sebelum dari triplet,apabila tidak terdapat maka dilakukan pendeteksian yang digambarkan pada baris kode ke 4 pada kode sumber 4.15. Pada baris kode ke 4 fungsi *checkSubyekNotPredikat* ditunjukkan pada potongan kode sumber 4.16 dan apabila terdapat maka dilakukan pendeteksian yang digambarkan pada baris kode ke 9 pada kode sumber 4.15.Pada baris kode ke 9 fungsi *checkSubyekYesPredikat* ditunjukkan pada potongan kode 4.17

```
1.  private String checkSubyekNotPredikat(String subyek,String predikat1,String obyek1)
    {
2.          String subyekMantap=null;
3.          tripletLoop:
4.          for(int y=0;y<isiKelasBaru.size();y++){
5.          ...
6.                      if(subyek.equalsIgnoreCase(namaKelas))
7.                      {
8.                          if(!isiMetode.isEmpty()){
9.                              for(int k=0;k<isiMetode.size();k++){
10.                                 ...
11.                                 if(namaKelas.equals(pemilikMetode)){
12.                                     ...
13.                                     for(int l=0;l<isiKodeSumber.size();l++){
14.                                         ...
15.                                         if(namaMetode.equals(pemilikKodeSumber)){
16.                                             ...
17.                                                 if(pemanggilKode.equalsIgnoreCase(pred
    ikat1) && dipanggilKode.equalsIgnoreCase(obyek1)){
18.                                                     subyekMantap = subyek+" (V) ";
19.                                                     predikatSebelum = predikat1;
20.                                                     obyekSebelum = obyek1;
21.                                                     break tripletLoop;  }}}}}}}
22.                     else
23.                     {
24.                         subyekMantap = subyek+" (X) ";}}
25.         return subyekMantap;}
```

***Kode Sumber 4.16 Potongan kode fungsi checksSubyekNotPredikat***

```
01. private String checkSubyekYesPredikat(String subyek,String predikat1,String obyek1){
02.         String subyekMantap=null;
03.         tripletLoop:
04.            for(int k=0;k<isiMetode.size();k++){
05.              String namaMetode = isiMetode.get(k).getNamaMetode();
06.              String pemilikMetode = isiMetode.get(k).getPemilik();
07.                       if(subyek.equalsIgnoreCase(pemilikMetode)){
08.                  for(int l=0;l<isiKodeSumber.size();l++){
09.                     String pemilikKodeSumber = isiKodeSumber.get(l).getPemilik();
10.                     String pemanggilKode = isiKodeSumber.get(l).getPemanggil();
11.                     String dipanggilKode = isiKodeSumber.get(l).getDipanggil();
12.                       if(obyekSebelum.equals(pemilikKodeSumber)){
13.                         System.out.println(namaMetode+" "+pemilikKodeSumber);
14.                         String simpanKodeSumber = isiKodeSumber.get(l).toString();
15.                           if(pemanggilKode.equalsIgnoreCase(predikat1) && dipanggilKode.equalsIgnoreCase(obyek1)){
16.                              subyekMantap = subyek+" (V) ";
17.                              predikatSebelum = predikat1;
18.                              obyekSebelum = obyek1;
19.                              break tripletLoop;
20.                             }
21.                           }
22.                         else{
23.                          subyekMantap = subyek+" (X) ";
24.                           }
25.                }
```

***Kode Sumber 4.17 Potongan kode fungsi checkSubyekYesPredikat***

Dari implementasi diatas maka akan didapatkan sebuah hasil deteksi yang berupa triplet bersama hasil deteksi berupa penanda pada setiap subyek, predikat,dan obyek sesuai dengan penjelasan pada bab 3.2.2.7.

### 4.2.2.4 Implementasi Word Similarity

Setelah berhasil melakukan pendeteksian,dilanjutkan dengan melakukan perhitungan word similarity terhadap metode yang ada pada kode sumber dan predikat yang terdapat pada triplet.Hal ini dijelaskan pada sub-bab 3.2.2.8.Dari penjelasan pendeteksian dijelaskan melalui contoh kasus yang dijelaskan dibawah ini:
Terdapat sebuah triplet diagram urutan yang meliputi :
1. user, input, MainPage
2. MainPage, register, RegisterController
3. RegisterController, createAccount, Account
4. RegisterController, setId, Account
5. RegisterController, setName, Account
6. RegisterController, setAge, Account
7. RegisterController, add, List
8. List, showSuccess, MainPage

Dan kode sumber yang terbentuk adalah
Nama Kelas : public class Account
List Metode :
1. public void setId
2. public null getId
3. public void setName
4. public String getName
5. public void setAge
6. public String getAge

Dari triplet diatas dan kode sumber yang terdapat dilakukan pendeteksian *similarity* terhadap predikat dan daftar metode dalam sebuah kelas.Hal tersebut dilakukan setiap iterasi triplet.Pada kasus ini didapati iterasi triplet dengan predikat *createAccount*(pada urutan triplet ke-3) dilakukan perhitungan word similarity melalui wordnet dengan perhitungan wu-palmer terhadap metode didalam kode sumber.selanjutnya *createAccount* dilakukan iterasi perhitungan terhadap masing-masing metode yang terdapat dalam kelas *account*.Didalam kelas *account* terdapat satu metode yang memilikki similarity dengan *tresshold* yang tinggi,perhitungan tersebut dilakukan dengan membandingkan *createAccount* dengan *setName*.Proses dimulai dengan melakukan *preProcessing* terhadap obyek dan metode ini,sehingga *createAccount* menjadi 2 buah kata yaitu *create* dan *account*,Metode *preProcessing* ini digambarkan pada kode sumber 4.18.

```
1.  public ArrayList<String> splitCamelCaseString(String s){
2.      ArrayList<String> result = new ArrayList<String>();
3.      for (String w : s.split("(?<!(^|[A-Z]))(?=[A-Z])|(?<!^)(?=[A-Z][a-z])")) {
4.          result.add(w);
5.      }
6.      return result;}
```

**Kode Sumber 4.18 Fungsi pemisahan metode dan obyek**

Sedangkan untuk *setName* menjadi dua buah kata yaitu set dan name.Masing-masing dari kata tersebut dihitung *similarity*-nya.hasil similarity dibentuk melalui tabel dibawah ini.

**Tabel 4.1 tabel perbandingan similarity**

| Persamaan Kata | Nilai Similarity |
|---|---|
| create set | 0.40 |
| Create-name | 0.33 |
| Account-set | 0.42 |
| Account-Name | 0.4 |

Nilai tabel diatas didapatkan setelah selesai melakukan pemisahan *pre-processing* hasil tersebut dimasukkan ke dalam fungsi *compute* yang digambarkan melalui kode sumber 4.19

```
1.  public double[][] getSimilarityMatrix( String[] words1, String[] words2, Relatednes
    sCalculator rc ){
2.     double[][] result = new double[words1.length][words2.length];
3.     for ( int i=0; i<words1.length; i++ ){
4.         for ( int j=0; j<words2.length; j++ ) {
5.             double score = rc.calcRelatednessOfWords(words1[i], words2[j]);
6.             result[i][j] = score;
7.                if(result[i][j]>1.00){
8.                     result[i][j] = 1.00;
9.                 }
10.          }
11.        }
12.    return result;}
13.
```

**Kode Sumber 4.19 Fungsi compute similarity**

Kode sumber 4.19 mendapatkan *input* dari hasil *pre-processing*, lalu setiap kata dari tabel dimasukkan ke dalam variable *words1* dan *words2* untuk dilakukan perhitungan similaritty.Perhitungan similarity ditunjukkan pada baris ke 5 pada kode sumber 4.19.

Perhitungan dilakukan dengan menggunakan *calcRelatednessOfwords* yang merupakan perhitungan similarity dari wu-palmer yang telah dijelaskan sebelumnya. Setelah hasil similarity didapatkan, hasil kemudian diproses untuk dihitung hasil rata-rata dari perhitung similarity ini yang ditunjukkan pada kode sumber 4.20. Nantinya hasil dari similarity kata yang digambarkan pada tabel 4.1 dihitung nilai kesimpulan similaritynya.

```
1.  public double Averagearray(double[][] array) {
2.      double total=0;
3.      int totallength=0;
4.              for(int i=0;i<array.length;i++) {
5.              for(int j=0;j<array[i].length;j++) {
6.                  total+=array[i][j];
7.                  totallength++;
8.              }
9.          }
10.     return total/(totallength);}
```

**Kode Sumber 4.20 Fungsi perhitungan kesimpulan similarity**

Dari kode sumber 4.20 nantinya ditemukan hasil dari total dari *similarity* yang nantinya digunakan untuk menjadi hasil yang menentukan kedua kata tersebut memilikki kesamaan atau tidak.Dari proses ini didapatkan hasil similarity adalah 0.39. Hasil ini memenuhi treshold dari kakas bantu yang ditetapkan pada angka 0.35, dari hasil tersebut didapatkan bahwa *createAccount* dan *setName* memilikki *similarity* yang sama dan dapat ditandai sebagai benar di dalam hasil kakas bantu.

#### 4.2.2.5 Implementasi Ekspor hasil deteksi

Setelah proses pendeteksian sudah selesai dilakukan analis sistem diberi pilihan untuk melakukan ekspor hasil deteksi berkas ke ".pdf". Proses ini dimulai ketika analis menekan tombol ekspor.

Untuk implementasi ini dimulai saat fungsi *writePdf* dipanggil. Fungsi *writePdf* ditunjukkan pada kode sumber 4.21.

```
1.  public class PDFwriter {
2.      public static void writePdf(String filename,String hasilEkstraksiDiagram,String
     hasilEkstraksiKode,String hasilDeteksi) throws DocumentException{
3.          Document document = new Document();
4.          try {
5.              System.out.println("DARI PDFWRITER"+hasilDeteksi);
6.              PdfWriter.getInstance(document, new FileOutputStream(new File(filename)
    ));
7.              //mulai dengan membuka PDF
8.              document.open();
9.              Font f = new Font();
10.             f.setSize(16);
11.             f.setStyle(Font.BOLD);
12.             document.add(new Paragraph("Hasil Ekstraksi diagram urutan",f));
13.             //isi dari hasil ekstraksi dimasukkan ke paragraph selanjutnya
14.             Paragraph p2 = new Paragraph();
15.             p2.add(hasilEkstraksiDiagram);
16.             document.add(p2);
17.             document.add(new Paragraph("Hasil Ekstraksi Kode Sumber",f));
18.             //isi dari hasil ekstraksi dimasukkan ke paragraph selanjutnya
19.             Paragraph p3 = new Paragraph();
20.             p3.add(hasilEkstraksiKode);
21.             document.add(p3);
22.             document.add(new Paragraph("Hasil Deteksi: ",f));
23.             Paragraph p4 = new Paragraph();
24.             p4.toString();
25.             p4.add(hasilDeteksi);
26.             document.add(p4);
27.             document.close();
28.             System.out.println("Berhasil convert PDF");
29.             } catch (FileNotFoundException ex) {
30.             System.out.println("error: "+ex.getMessage());
31.         }}}
```

***Kode Sumber 4.21 Potongan kode fungsi ekspor hasil pendeteksian***

Fungsi *writePdf* memilikki 3 buah parameter yaitu *hasilEkstraksiDiagram, hasilEkstraksiKode, dan hasilDeteksi*. Ketiga itu didapatkan dari fungsi *getText* yang didapatkan dari *textArea* dari komponen antar muka yang sebelumnya sudah

diimplementasi 4.2.1.1. Ekspor hasil deteksi ini dibantu oleh pustaka *ItextPdf* yang ditunjukkan pada baris kode 5 pada kode sumber 4.21. Setelah itu penulisan hasil deteksi ke dalam PDF terdapat pada baris kode ke 15 dengan menambahkan paragraph yang sudah di *instance* sebelumnya pada baris kode ke 14 ke dalam dokumen.Setelah itu penulisan masing-masing hasil yang didapatkan dari parameter kemudian dilakukan.

***(Halaman ini sengaja dikosongkan)***

# BAB V
# PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tentang pengujian dan evaluasi pada kakas bantu yang telah dibuat. Pengujian ini dilakukan dengan mencari diagram urutan beserta kode sumber dari diagram urutan tersebut. Evaluasi berisi penjabaran dari hasil masing-masing pengujian

## 5.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan dengan menggunakan perangkat bantu dilakukan dengan menggunakan komputer dengan spesifikasi Intel® Core™ i7-4700HQ @2.40GHz dengan memori sebesar 4GB.

## 5.2 Data Pengujian

Data pengujian yang digunakan untuk menguji hasil deteksi ketidaksesuaian diagram urutan dan kode sumber dari sebuah kasus perangkat lunak yang diambil dari internet. Data yang didapatkan berupa diagram urutan dan kode sumber berbasis java. Masing-masing harus diestraksi terlebih dahulu,untuk diagram urutan diesktraksi dengan menggunakan *StarUml2*. Sedangkan untuk kode sumber diekstraksi dengan menggunakan *xmltranslator* yang terdapat di internet. Total dataset dari diagram urutan ini sebanyak 5 buah kasus diagram urutan.

## 5.3 Skenario Pengujian

Skenario pengujian terdiri dari tiga bagian, yaitu skenario pengujian hasil deteksi ketidaksesuaian diagram urutan dengan kode sumber,skenario pengujian fungsional kakas bantu,dan skenario pengujian non-fungsional kakas bantu.Pertama, pengujian hasil deteksi ketidaksesuaian dilakukan dengan menggunakan 5 buah kasus diagram urutan dengan sebanyak 10

pasang kasus diagram dan kode sumber.Ada tiga skenario pengujian hasil ketidaksesuaian.skenario tersebut,yaitu pengujian hasil deteksi ketidaksesuaian sesuai dengan teori analisis diagram urutan terhadap kode sumber,dan pengujian hasil deteksi ketidaksesuaian dibandingkan terhadap pengujian manual yang dilakukan oleh ahli rekayasa perangkat lunak.Untuk hasil deteksi ketidaksesuaian sesuai teori analisis diagram urutan terhadap kode sumber berfungsi untuk mengetahui apakah kakas bantu yang dibuat sudah sesuai dengan teori tersebut.Untuk hasil pendeteksian ketidaksesuaian yang melibatkan ahli,hasil deteksi oleh kakas bantu dan ahli dibandingkan dengan menghitung secara hitungan *statistik*.

Kedua, pengujian fungsional kakas bantu.Pengujian fungsional kakas bantu dilakukan menggunakan pengujian *black-box*. Pengujian *black-box* adalah pengujian yang menguji fungsionalitas perangkat lunak untuk menenteukan fungsionalitas tersebut sudah memenuhi dengan apa yang diharapkan.Pengujian ini nantinya dilakukan dengan menyiapkan sebuah skenario yang merujuk kepada kasus penggunaan(*use case*) tersebut. Kemudian partisipan diberikan tugas untuk menjalankan skenario tersebut.Skenario pengujian ini terdiri dari lima pengujian fitur utama yaitu pengujian membuat proyek,pengujian menyimpan proyek,dan pengujian mengekspor proyek,pengujian membuka proyek, dan pengujian memeriksa ketidaksesuaian. Hasil pengujian fungsional dari setiap skenario diatas ditentukan dengan membandingkan hasil yang didapatkan dengan hasil yang diharapkan.

Ketiga, pengujian non fungsional kakas bantu.Pengujian non-fungsional dilakukan untuk mengetahui kemudahan penggunaan dan kesesuaian penggunaan kakas bantu.Di awal,partisipan telah diberikan penjelasan mengenai cara kerja kakas bantu.Setelah partisipan selesai menjalankan kakas bantu partisipan diminta untuk mengisi kuisioner.

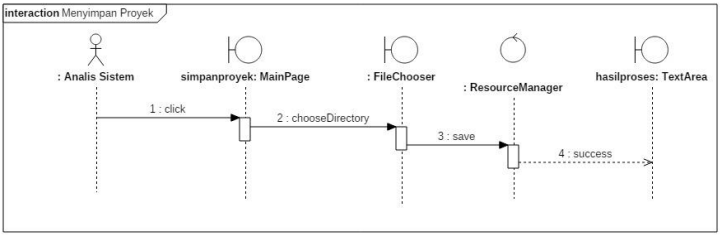
### 5.3.1 Pengujian Hasil Deteksi Ketidaksesuaian

Pengujian hasil deteksi ketidaksesuaian dilakukan dengan skenario yang telah dijelaskan pada skenario pengujian. Untuk setiap pengujian hasil deteksi akan dijelaskan pada masing-masing sub-bab berikut ini.

#### 5.3.1.1 Pengujian hasil deteksi ketidaksesuaian kakas bantu sesuai teori analisis kode sumber terhadap diagram urutan

Kode sumber dibuat berdasarkan alur yang ada pada diagram urutan. alur pada diagram urutan digambarkan dengan pesan(*message*) yang dikirimkan dan diterima baik oleh lifeline yang menggambarkan aktor atau kelas. Tahapan selanjutnya adalah menganalisis bagian dari diagram urutan untuk ketidaksesuaian subyek, predikat, dan obyek . Subyek sesuai apabila subyek merupakan aktor, subyek terdapat didalam kelas didalam kode sumber, dan didalam metode kelas subyek terdapat pemanggilan kode sumber yang mengarah ke obyek. predikat sesuai apabila predikat merupakan metode didalam kelas obyek. Obyek sesuai apabila obyek merupakan kelas yang terdapat dalam kode sumber.

Berdasarkan teori diatas skenario pengujian ini dilakukan untuk mengetahui apakah kakas bantu yang sudah dibuat sudah memenuhi teori tersebut. Pengujian digunakan dengan menggunakan sebuah diagram urutan "*save function*" pada kakas bantu ini. Berikut adalah Diagram urutan dari kasus ini ditunjukkan pada gambar 5.1

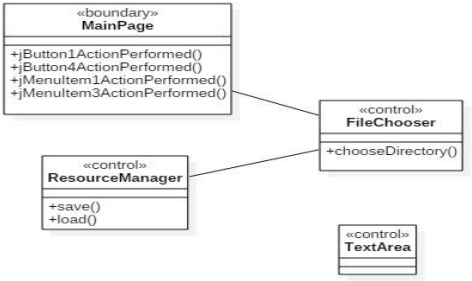



**Gambar 5.1 Gambar kasus diagram urutan "Save function"**

Dari diagram urutan diatas didapatkan triplet yang dihasilkan berupa :

1. Analis Sistem, click, MainPage
2. MainPage, chooseDirectory, FileChooser
3. FileChooser, save, ResourceManager
4. ResourceManager, success, TextArea

Sedangkan untuk kode sumber digambarkan melalui kelas diagram, yang ditunjukkan pada gambar 5.2




**Gambar 5.2 Gambar kasus Diagram kelas "save function"**

Dan dari kelas diagram tersebut dibentuk kode sumber yang digambarkan dibawah ini berturut-turut dari gambar 5.3.

```
01. public class MainPage extends javax.swing.JFrame {
02.     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
03.         fc.chooseDirectory(pathDiagramUrutan,pathKodeSumber ,hasilDiagramUrutan , hasilKodeSumber,hasilDeteksi );
04.     }
05.
06.     private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
07.         PDFwriter pw = new PDFwriter();
08.         JFileChooser jfPDF = new JFileChooser();
09.         FileNameExtensionFilter jfOpenFilter = new FileNameExtensionFilter("PDF Files","pdf");
10.         int value = jfPDF.showSaveDialog(null);
11.         if(value == JFileChooser.APPROVE_OPTION){
12.             try {
13.                 File pdfPath = jfPDF.getSelectedFile();
14.                 pw.writePdf(pdfPath.getPath()+".pdf", hasilDiagramUrutan, hasilKodeSumber, hasilDeteksi);
15.
16.             } catch (DocumentException ex) {
17.                 Logger.getLogger(MainPage.class.getName()).log(Level.SEVERE, null, ex);
18.             }
19.         }
20.     }
21.
22.     private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
23.           newProyek();
24.     }
25.
26.     private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
27.
28.     }
29. }
```

**Gambar 5.3 Gambar Kode sumber MainPage**

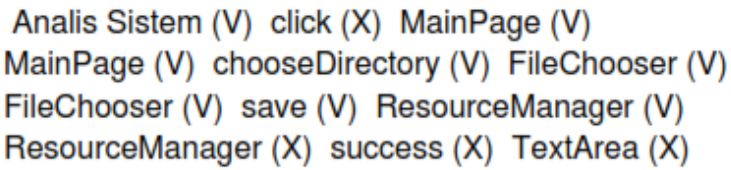
```
01. public class ResourceManager {
02.     public static void save(Serializable data,String fileName)throws Exception{
03.
04.     }
05.
06.     public static Object load(String fileName)throws Exception{
07.
08.     }
09. }
```

**Gambar 5.4 Gambar Kode sumber ResourceManager**

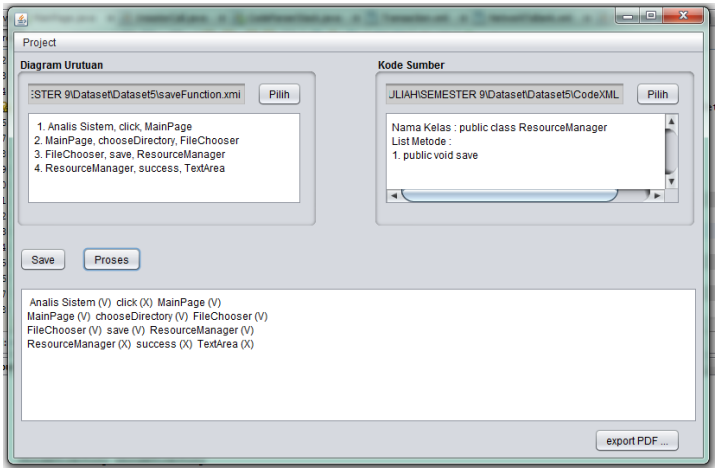
```
01. public class FileChooser {
02.
03.    public void chooseDirectory(String pathdiagramUrutan,String pathKode,String hasilDiagramUrutan,String hasilKodeSumber,String hasilDeteksi){
04.        JFileChooser jfSave = new JFileChooser();
05.        FileNameExtensionFilter jfSaveFilter = new FileNameExtensionFilter("File Project Pendeteksian Diagram Urutan","pkdu");
06.        ResourceManager rc = new ResourceManager();
07.        if(value == JFileChooser.APPROVE_OPTION){
08.            SaveData data = new SaveData();
09.            rc.save(data, fi.getPath()+".pkdu");
10.        }
11.
12.    }
13. }
```

**Gambar 5.5 Gambar Kode sumber FileChooser**

Dari teori diatas didapatkan hasil dari pendeteksian berupa berikut,ditunjukkan pada gambar 5.6




**Gambar 5.6 Gambar Hasil deteksi ketidaksesuaian teori**




**Gambar 5.7 Gambar Hasil deteksi Kakas Bantu**

Dari gambar 5.7 didapatkan pada triplet pertama, menggambarkan bahwa subyek mendapatkan benar,karena subyek memenuhi karena merupakan aktor,sedangkan predikat tidak memenuhi karena merupakan gambaran yang dilakukan dari aktor ke dalam sistem kakasbantu,sedangkan untuk *MainPage* memenuhi karena terdapat didalam kelas dari hasil ekstraksi kode sumber.Lalu dari triplet bagian kedua didapatkan subyek benar dikarenakan didalam

subyek *MainPage* merupakan sebuah kelas,dan terdapat pemanggilan kode sumber *FileChooser.chooseDirectory* didalam metode kelas subyek,maka dari itu subyek memenuhi.Sedangkan untuk predikat memenuhi karena predikat terdapat didalam kelas obyek.Sedangkan Obyek memenuhi karena obyek FileChooser terdapat didalam kelas hasil ekstraksi kode sumber.Lalu dari triplet ke tiga subyek memenuhi karena didapatkan Subyek *FileChooser* terdapat didalam kelas hasil ekstraksi kode sumber,dan terdapat pemanggilan kode sumber *ResourceManager.save* didalam metode kelas subyek.Sedangkan predikat memenuhi karena predikat merupakan metode bernama *save* terdapat didalam kelas obyek.Kelas Obyek memenuhi karena Obyek terdapat dalam kelas hasil ekstraksi kode sumber.Untuk triplet terakhir tidak memenuhi semua dikarenakan triplet ini tidak terdapat di kode sumber dikarenakan triplet ini menggambarkan result sukses didalam diagram urutan.Dari penjelasan diatas didapatkan bahwa kakas bantu sudah dibentuk sesuai dengan teori yang ada.

#### 5.3.1.2 Pengujian Hasil Deteksi Ketidaksesuaian antara Kakas Bantu dan Ahli

Sesuai penjelasan pada bab 2 sub-bab 2.8,terdapat pengujian hasil deteksi ketidaksesuaian antara kakas bantu dan ahli rekayasa perangkat lunak.Pengujian menggunakan diagram urutan sesuai dengan penjelasan yang telah dijelaskan pada sub bab data pengujian.Pengujian tersebut dijabarkan kedalam 5 buah kasus diagram urutan beserta kode sumber.Hasil dari percobaan dari masing-masing kasus dijabarkan sebagai berikut:

**1. Kasus register management Library**

Dari 8 buah triplet diagram urutan dan kode sumber pada fungsi register pada sistem MANAGEMENT LIBRARY, kakas bantu memberikan sebanyak 4 buah triplet yang “sesuai”. Dan sebanyak

4 buah triplet diberikan label “tidak sesuai” oleh kakas bantu, namun “sesuai” oleh ahli. Sedangkan untuk ahli memberikan label 7 triplet sesuai sedangkan 1 tidak sesuai.

Kakas bantu dan ahli memberikan label “sesuai” sebanyak 4 pasang, dan 3 pasang diberikan label “tidak sesuai” oleh kakas bantu,namun “sesuai” oleh ahli . Dan tidak ditemukan label “sesuai” namun “tidak sesuai” oleh ahli pada kakas bantu. Selanjutnya terdapat 1 label yang diberikan label “tidak sesuai” pada keduanya. Penjelasan tersebut digambarkan pada tabel 5.1

**Tabel 5.1 Hasil deteksi antara kakas bantu dan ahli pada kasus register management library**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 4 | 3 | 7 |
| **Tidak sesuai** | 0 | 1 | 1 |
| **Jumlah** | 4 | 4 | 8 |

Dalam perhitungan kappa statistic ,hasil percobaan diatas menghasilkan nilai kesepakatan 0,33.

**2. Kasus register Management Library dan kode sumber berbeda**

Dari 8 buah triplet diagram urutan sumber pada fungsi register pada sistem MANAGEMENT LIBRARY dan kode sumber ATM. kakas bantu memberikan sebanyak 0 buah label yang “sesuai”. Dan sebanyak 8 buah triplet diberikan label “tidak sesuai” oleh kakas bantu . Sedangkan untuk ahli memberikan label 0 sesuai sedangkan 8 tidak sesuai. Penjelasan digambarkan melalui tabel 5.2

**Tabel 5.2 Hasil deteksi antara kakas bantu dan ahli pada kasus register management library dengan kode sumber berbeda**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 0 | 0 | 0 |
| **Tidak sesuai** | 0 | 8 | 8 |
| **Jumlah** | 0 | 8 | 8 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*

### 3. Kasus Shutdown pada ATM

Terdapat 3 buah pasang triplet dari diagram urutan kasus *shutdown* pada kasus ini.untuk percobaan ini ahli perangkat lunak dan kakas bantu memberi 1 buah label "sesuai". Lalu tidak ditemukan label "sesuai" dari ahli dan "tidak sesuai" dari kakas bantu. Lalu tidak ditemukan label "sesuai" dari kakas bantu tetapi "tidak sesuai" dari ahli. Selanjutnya ditemukan 2 buah label yang "tidak sesuai" dari kakas bantu dan "tidak sesuai" daru ahli. Penjelasan digambarkan melalui tabel 5.3

**Tabel 5.3 Hasil pendeteksian antara kakas bantu dan ahli pada kasus Shutdown pada ATM**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 1 | 0 | 1 |
| **Tidak sesuai** | 0 | 2 | 2 |
| **Jumlah** | 1 | 2 | 3 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

**4. Kasus Shutdown pada ATM dan Kode sumber berbeda**

Terdapat 3 buah pasang triplet dari diagram urutan kasus *shutdown* pada kasus ini. Untuk percobaan ini ahli perangkat lunak dan kakas bantu tidak menemukan label "sesuai", selanjutnya tidak ditemukan juga label yang "tidak sesuai" dari kakas bantu dan "sesuai" dari ahli. Untuk label "tidak sesuai" dari ahli dan "sesuai" dari kakas bantu tidak ditemukan juga. Namun ahli perangkat lunak dan kakas bantu sama-sama menemukan 3 triplet "tidak sesuai". Penjelasan digambarkan melalui tabel 5.4

**Tabel 5.4 Hasil pendeteksian antara kakas bantu dan ahli pada kasus shutdown dengan kode sumber berbeda**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 0 | 0 | 0 |
| **Tidak sesuai** | 0 | 3 | 3 |
| **Jumlah** | 0 | 3 | 3 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

**5. Kasus Save pada aplikasi kakas banntu ini**

Terdapat 4 buah pasang triplet dari diagram urutan kasus *save* pada kasus ini. Untuk percobaan ini ahli perangkat lunak dan kakas bantu menemukan 2 buah label yang "sesuai". Sedangkan tidak ditemukan label yang "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli perangkat lunak. Dan tidak ditemukan label yang "sesuai" oleh kakas bantu namun "tidak sesuai" oleh ahli perangkat lunak. Selanjutnya ditemukan 2 buah label "tidak sesuai" oleh ahli dan kakas bantu. Penjelasan digambarkan melalui tabel 5.5

**Tabel 5.5 Hasil pendeteksian antara kakas bantu dan ahli pada kasus save aplikasi kakas bantu**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 2 | 0 | 2 |
| **Tidak sesuai** | 0 | 2 | 2 |
| **Jumlah** | 2 | 2 | 4 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

### 6. Kasus save pada aplikasi kakas bantu dan kode sumber berbeda

Terdapat 4 buah pasang triplet dari diagram urutan kasus *save* ini. Untuk percobaan ini ahli perangkat lunak dan kakas bantu tidak menemukan triplet yang "sesuai". Selanjutnya tidak ditemukan label yang dideteksi "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli. Tidak ditemukan juga label "sesuai" dari kakas bantu namun "tidak sesuai" oleh ahli.Selanjutnya ahli dan kakas bantu menemukan 4 buah label yang "tidak sesuai". Penjelasan digambarkan melalui tabel 5.6

**Tabel 5.6 Hasil pendeteksian antara kakas bantu dan ahli pada kasus save dengan kode sumber yang berbeda**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 0 | 0 | 0 |
| **Tidak sesuai** | 0 | 4 | 4 |
| **Jumlah** | 0 | 4 | 4 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

**7. Kasus Transaction pada ATM**

Terdapat 8 buah pasang triplet dari diagram urutan kasus *transaction* ini. Untuk percobaan ini ahli perangkat lunak dan kakas bantu menemukan 7 buah label "sesuai", selanjutnya terdapat 1 buah label yang diindikasi "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli. Selanjutnya tidak ditemukan label "sesuai" oleh kakas bantu namun "tidak sesuai" oleh ahli, dan tidak ditemukan juga triplet yang diidentifikasi "tidak sesuai" oleh kakas bantu dan ahli. Penjelasan digambarkan melalui tabel 5.7

**Tabel 5.7 Hasil pendeteksian antara kakas bantu dan ahli pada kasus transaction pada ATM**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 7 | 1 | 8 |
| **Tidak sesuai** | 0 | 0 | 0 |
| **Jumlah** | 7 | 1 | 8 |

Sesuai tabel tersebut,percobaan diatas menghasilkan nilai kesepakatan 0.85 melalui perhitungan *kappa statistic*.

**8. Kasus Transaction pada ATM dengan kode sumber berbeda**

Terdapat 8 buah pasang triplet dari diagram urutan kasus *transaction ini.* Untuk percobaan ini diagram urutan di cocokkan dengan kode sumber milik ATM. Didapati ahli perangkat lunak dan kakas bantu tidak menemukan label "sesuai", dan tidak ditemukan juga label "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli. Selanjutnya tidak ditemukan juga label "sesuai" oleh kakas bantu namun "tidak sesuai" oleh ahli dan ditemukan 8 buah label "tidak sesuai" yang diidentifikasi oleh kakas bantu dan ahli. Penjelasan tersebut digambarkan melalui tabel 5.8

**Tabel 5.8 Hasil pendeteksian kakas bantu dan ahli pada kasus transaction pada ATM dengan kode sumber berbeda**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 0 | 0 | 0 |
| **Tidak sesuai** | 0 | 8 | 8 |
| **Jumlah** | 0 | 8 | 8 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

**9. Kasus Start up pada ATM**

Terdapat 6 buah pasang triplet dari diagram urutan kasus *start up* ini. Untuk percobaan ini ahli perangkat lunak dan kakas bantu mendapatkan 2 buah label "sesuai", dan mendapatkan 1 buah label "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli. Selanjutnya ditemukan 1 buah label "sesuai" oleh kakas bantu namun "tidak sesuai" oleh ahli, dan 2 buah label "tidak sesuai" yang diidentifikasi oleh kakas bantu dan ahli. Penjelasan tersebut digambarkan melalui tabel 5.9

**Tabel 5.9 Hasil pendeteksian kakas bantu dan ahli kasus Start up pada ATM**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 2 | 1 | 3 |
| **Tidak sesuai** | 1 | 2 | 3 |
| **Jumlah** | 3 | 3 | 6 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 0.32 melalui perhitungan *kappa statistic*.

**10. Kasus Start up pada ATM dengan kode sumber berbeda**

Terdapat 6 buah pasang triplet dari diagram urutan kasus *start up* ini. Untuk percobaan ini kasus diagram urutan ini dibandingkan

terhadap kode dari fungsi *register* pada Kasus *management library*. Didapati ahli perangkat lunak dan kakas bantu tidak menemukan triplet "sesuai", dan tidak ditemukan juga triplet "tidak sesuai" oleh kakas bantu namun "sesuai" oleh ahli. Selanjutnya tidak ditemukan juga triplet "sesuai" oleh kakas bantu namun "tidak sesuai" oleh ahli dan ditemukan 6 buah triplet "tidak sesuai" yang diidentifikasi oleh kakas bantu dan ahli. Penjelasan tersebut digambarkan melalui tabel 5.10

**Tabel 5.10 Hasil pendeteksian kakas bantu dan ahli pada kasus Start up dengan kode sumber berbeda**

| **Ahli** | **Kakas Bantu** | | |
|---|---|---|---|
| | Sesuai | Tidak sesuai | Jumlah |
| **Sesuai** | 0 | 0 | 0 |
| **Tidak sesuai** | 0 | 6 | 6 |
| **Jumlah** | 0 | 6 | 6 |

Sesuai tabel tersebut, percobaan diatas menghasilkan nilai kesepakatan 1.00 melalui perhitungan *kappa statistic*.

Dari percobaan hasil deteksi ketidaksesuaian yang dilakukan, nilai *kappa statistic* setiap kasus antar kakas bantu dan ahli dijabarkan oleh tabel 5.11.

**Tabel 5.11 Hasil Percobaan Kappa Statistic**

| **Pengamat** | **Kasus** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Kakas Bantu - Ahli** | 0.33 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.85 | 1.00 | 0.32 | 1.00 |

Nilai kesepakatan tertinggi dari ahli dan kakas bantu terjadi pada sebanyak 7 buah kasus.

Sedangkan nilai kesepakatan terendah terdapat pada kasus 9, dengan nilai kesepakatan 0.32. apabila total dari 10 kasus tersebut

dihitung secara keseluruhan nilai kesepakatan didapatkan sejumlah 0.95.

### 5.3.2 Pengujian Fungsional

Sesuai dengan penjelasan sebelumnya, pengujian fungsional kakas bantu dilakukan melalui *black-box testing* dengan menyiapkan skenario sebagai tolak ukur keberhasilan. Adapun partisipan yang terlibat dalam pengujian ini adalah mahsiswa Teknik Informatika ITS dengan kualifikasi sudah lulus mata kuliah analisis perancangan sistem informasi dan perancangan perangkat lunak. Jumlah mahasiswa yang terlibat adalah 5 orang. Selanjutnya,pengujian ini dijabarkan sebagai berikut.

#### 5.3.2.1 Pengujian Fitur memeriksa ketidaksesuaian

Pengujian ini digunakan untuk menguji fitur memeriksa ketidaksesuaian apakah sudah sesuai dengan hasil yang diharapkan.Pengujian fitur ini terdiri dari sebuah skenario. Skenario pengujian dijabarkan oleh tabel 5.12

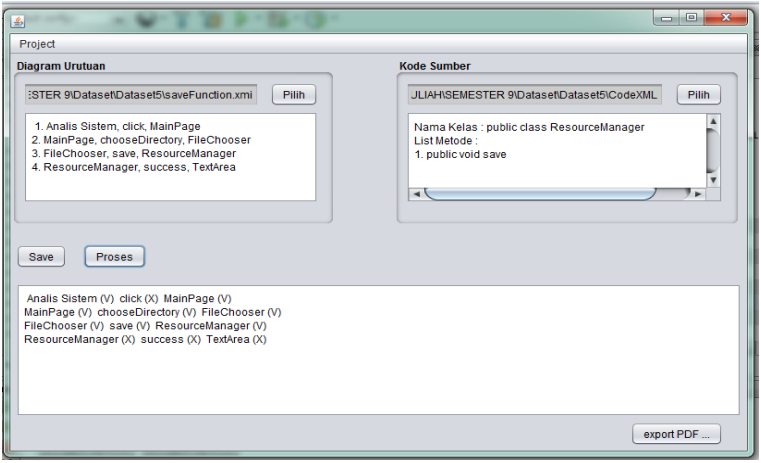
**Tabel 5.12 Skenario Pengujian fitur memeriksa ketidaksesuaian**

| | |
|---|---|
| **Nama Pengujian** | Pengujian fitur memeriksa ketidaksesuaian diagram urutan terhadap kode sumber |
| **Tujuan Pengujian** | Menguji fitur untuk memeriksa ketidaksesuaian diagram urutan terhadap kode sumber |
| **Refrensi Kasus penggunaan** | Memeriksa ketidaksesuaian |
| **Skenario** | *Partisipan memeriksa diagram urutan dengan memasukkan berkas dokumen rancangan berupa format ."xmi" dan memilih folder tempat dimana berkas kode sumber berbentuk ".xml" disimpan* |

**Tabel 5.12 Skenario pengujian fitur memeriksa ketidaksesuaian (*lanjutan*)**

| **Kondisi awal** | **Diagram urutan dan kode sumber yang belum diperiksa ketidaksesuaiaanya** |
|---|---|
| **Langkah pengujian** | Menekan tombol "pilih diagram urutan"<br>Memilih diagram urutan pada lokasi dimana diagram urutan disimpan<br>Menekan tombol "pilih kode sumber"<br>Memilih folder dimana kode sumber disimpan<br>Menekan tombol "proses" pada kakas bantu |
| **Hasil yang diharapkan** | Kakas bantu menampilkan hasil ekstraksi diagram urutan,hasil ekstraksi kode sumber dan hasil deteksi ketidaksesuaian pada layar kakas bantu |
| **Hasil yang didapatkan** | Kakas bantu menampilkan hasil ekstraksi diagram urutan, hasil ekstraksi kode sumber dan hasil deteksi ketidaksesuaian |
| **Kondisi Akhir** | Ketidaksesuaian pada triplet diagram urutan sesuai atau tidak sesuai |
| **Hasil pengujian** | Berhasil |

Hasil yang didapatkan pada skenario diatas yaitu berupa daftar triplet yang ditandai sebagai benar atau tidak. Daftar tersebut dapat dilihat pada gambar 5.8




**Gambar 5.8 Gambar hasil skenario Mendeteksi ketidaksesuaian**

### 5.3.2.2 Pengujian fitur ekspor hasil pemeriksaan ketidaksesuaian

Pengujian ini digunakan untuk menguji fitur mengekspor laporan hasil pemeriksaan ketidaksesuaian apakah sudah sesuai dengan hasil yang diharapkan. Pengujian fitur ini dijelaskan pada skenario dibawah ini, skenario dijabarkan oleh tabel 5.13

**Tabel 5.13 Skenario pengujian fitur Ekspor hasil pemeriksaan ketidaksesuaian**

| | |
|---|---|
| **Nama Pengujian** | Pengujian fitur mengekspor laporan pemeriksaan ketidaksesuaian |
| **Tujuan Pengujian** | Menguji fitur untuk mengekspor hasil pemeriksaan ketidaksesuaian ke dalam bentuk laporan berkas ".pdf" |
| **Refrensi Kasus penggunaan** | Mengekspor laporan hasil |
| **Skenario** | *Partisipan menyimpan lokasi dan nama berkas laporan hasil deteksi ketidaksesuaian ke dalam format ".pdf"* |

**Tabel 5.13 Skenario pengujian fitur ekspor hasil pemeriksaan ketidaksesuaian (*lanjutan*)**

| | |
|---|---|
| **Kondisi awal** | **Telah selesai melakukan proses pendeteksian** |
| **Langkah pengujian** | Menekan tombol "Ekspor PDF"<br>Memilih lokasi penyimpanan berkas ekspor<br>Memasukkan nama berkas<br>Menekan tombol "save" pada jendela "Ekspor PDF" |
| **Hasil yang diharapkan** | Kakas bantu mengekspor hasil laporan pada lokasi berkas laporan disimpan |
| **Hasil yang didapatkan** | Kakas bantu berhasil mengeskpor hasil laporan pada lokasi berkas laporan disimpan |
| **Kondisi Akhir** | Berkas ekspor laporan disimpan ke folder yang telah ditentukan |
| **Hasil pengujian** | Berhasil |

### 5.3.2.3 Pengujian Menyimpan proyek ketidaksesuaian

Pengujian ini digunakan untuk menguji fitur menyimpan proyek ketidaksesuaian apakah sudah sesuai dengan hasil yang diharapkan. Pengujian fitur ini dijelaskan pada skenario dibawah ini, Skenario dijabarkan oleh tabel 5.14

**Tabel 5.14 Skenario pengujian fitur menyimpan proyek ketidaksesuaian**

| | |
|---|---|
| **Nama Pengujian** | Pengujian fitur Menyimpan proyek ketidaksesuaian |
| **Tujuan Pengujian** | Menguji fitur untuk menyimpan proyek ketidaksesuaian |
| **Refrensi Kasus penggunaan** | Menyimpan proyek |
| **Skenario** | *Partisipan menyimpan lokasi dan nama proyek* |
| **Kondisi awal** | Telah selesai membuat proyek |

**Tabel 5.14 Skenario pengujian fitur menyimpan proyek ketidaksesuaian**

| | |
|---|---|
| **Langkah pengujian** | Menekan tombol “Save”<br>Memilih lokasi simpan Proyek<br>Memasukkan nama berkas proyek<br>Menekan tombol “save” pada jendela “save” |
| **Hasil yang diharapkan** | Kakas bantu menyimpan hasil proyek ke dalam bentuk ekstensi “.pkdu” |
| **Hasil yang didapatkan** | Kakas bantu berhasil menyimpan proyek dengan ekstensi “.pkdu” ke dalam lokasi yang telah ditentukan |
| **Kondisi Akhir** | Berkas dengan ekstransi “.pkdu” disimpan ke dalam lokasi yang telah ditentukan |
| **Hasil pengujian** | Berhasil |

### 5.3.2.4 Pengujian membuka proyek

Pengujian ini digunakan untuk menguji fitur membuka proyek ketidaksesuaian apakah sudah sesuai dengan hasil yang diharapkan. Pengujian fitur ini dijelaskan pada skenario dibawah ini, skenario dijabarkan oleh tabel 5.15

**Tabel 5.15 Skenario pengujian fitur membuka proyek**

| | |
|---|---|
| **Nama Pengujian** | Pengujian fitur membuka proyek |
| **Tujuan Pengujian** | Menguji fitur untuk membuka proyek |
| **Refrensi Kasus penggunaan** | Membuka proyek |
| **Skenario** | *Partisipan Membuka proyek yang sudah di simpan sebelumnya dengan ekstensi “.pkdu”* |
| **Kondisi awal** | Telah membuka Kakas bantu dan memilikki file ekstensi “.pkdu” yang telah disimpan sebelumnya |

**Tabel 5.15 Skenario pengujian fitur membuka proyek**

| | |
|---|---|
| Langkah pengujian | Menekan menu "Membuka proyek" pada pilihan "project" pada *menu bar*<br>Memilih berkas dengan ekstensi ".pkdu"<br>Menekan tombol open |
| **Hasil yang diharapkan** | Kakas bantu menampilkan hasil yang disimpan sebelumnya |
| **Hasil yang didapatkan** | Kakas bantu berhasil menampilkan hasil pendeteksian yang disimpan sebelumnya |
| **Kondisi Akhir** | Berkas ".pkdu" yang disimpan sebelumnya berhasil dibuka |
| **Hasil pengujian** | Berhasil |

### 5.3.3 Pengujian Non Fungsional

Pengujian non-fungsional kakas bantu dilakukan untuk aspek *usability*. Untuk menguji aspek tersebut partisipan dilibatkan. Selama partisipan menjalankan pengujian ,partisipan melakukan eksplorasi dan memahami kerja kakas bantu.

Kuisioner yang diisi oleh partisipan terdiri dari dua bagian, bagian pertama merupakan bagian pernyataan mengenai kakas bantu dan bagian kedua merupakan bagian pertanyaan mengenai kakas bantu. Bagian yang berisi daftar pernyataan terdiri dari 5 buah pernyataan.

1. Kakas bantu mudah digunakan
2. Kakas bantu dapat digunakan tanpa bantuan orang lain
3. Tampilan kakas bantu jelas
4. Kakas Menampilkan hasil deteksi ketidaksesuaian yang mudah dipahami
5. Kakas bantu sangat membantu apabila digunakan untuk mendeteksi ketidaksesuaian diagram urutan terhadap kode sumber.

Daftar pernyataan diisi dengan memberi nilai dari sangat tidak setuju ,tidak setuju,netral,setuju,dan sangat setuju. Masing-masing nilai tersebut diberikan bobot yang berbeda yang nantinya digunakan untuk menghitung persentase setiap pernyataan. Bobot 1 untuk sangat tidak setuju, 2 untuk tidak setuju, 3 untuk netral, 4 untuk setuju, dan 5 untuk sangat setuju. Nilai tersebut kemudian dijumlahkan dan dibagi dengan nilai tertinggi penilaian yaitu 25 (bobot untuk nilai tertinggi yaitu sangat setuju).Hasil perhitungan kuisioner ditunjukkan pada tabel 5.16

**Tabel 5.16 Hasil kuisioner**

| Pernyataan | Persentase |
|---|---|
| Kakas bantu mudah digunakan | 92% |
| Kakas bantu dapat digunakan tanpa bantuan orang lain | 72% |
| Tampilan kakas bantu jelas | 92% |
| Kakas bantu menampilkan hasil deteksi ketidaksesuaian yang mudah dipahami | 84% |
| Kakas bantu sangat membantu apabila digunakan untuk mendeteksi ketidaksesuaian diagram urutan terhadap kode sumber | 88% |

## 5.4 Evaluasi Pengujian

Sub bab ini menjelaskan analisis dan evaluasi dari skenario pengujian yang telah dilakukan sebelumnya. Analisis dan evaluasi dari skenario pengujian yang telah dilakukan dijelaskan sebagai berikut

### 5.4.1 Evaluasi pengujian hasil Deteksi Ketidaksesuaian

Pengujian hasil deteksi ketidaksesuaian yang telah dilakukan terdiri dua skenario yaitu pengujian hasil deteksi ketidaksesuaian sesuai dengan teori dan pengujian hasil deteksi ketidaksesuaian antara kakas bantu dan ahli. Berdasarkan hasil uji pengujian deteksi ketidaksesuaian sesuai teori pembuatan kode sumber terhadap diagram urutan, didapati kakas bantu yang dibuat dapat mendeteksi ketidak sesuaian diagram urutan dengan kode sumber. hal tersebut dapat ditunjukkan pada sub bab 5.3.1.1 yang menunjukkan hasil dari ketidaksesuaiaan kakas bantu yang sesuai dengan teori penulisan kode sumber terhadap diagram urutan.

Untuk pengujian hasil deteksi ketidaksesuaian antara kakas bantu dan ahli, nilai kesepakatan tertinggi terjadi pada sebanyak 7 buah kasus, hal tersebut terjadi karena kode sumber yang didapatkan sesuai terhadap diagram urutan yang ada. Dan menunjukkan bahwa proses ketidaksesuaian bisa berjalan dengan baik. Namun masih didapatkan beberapa kasus yang mendapatkan hasil yang rendah, hasil itu didapati pada kasus 9 dan 10 yang mendapatkan nilai 0.32 dan 0.33 hal itu disebabkan terdapat beberapa bagian pemanggilan atau triplet yang tidak dapat dideteksi oleh kakas bantu yang menyebabkan berkurangnya kualitas hasil dari kakas bantu.

Apabila semua kasus dihitung secara keseluruhan nilai kesepakatan antara kakas bantu dan ahli

didapatkan nilai kesepakatannya sejumlah 0.95. Nilai tersebut masuk ke kategori *almost perfect* menurut skala yang telah ditentukan oleh *Landis and koch* [2]

### 5.4.2 Evaluasi Pengujian Fungsional

Hasil Pengujian fungsional dirangkum oleh tabel 5.18 Dari semua skenario yang dilakukan uji coba, keseluruhannya berhasil berdasarkan tabel tersebut. Sehingga bisa ditarik kesimpulan, fungsionalitas dari kakas bantu yang dibuat telah sesuai dengan yang diharapkan.

**Tabel 5.17 Hasil Pengujian Fungsional**

| **Nama Fitur yang Diuji** | **Hasil** |
|---|---|
| Memeriksa Ketidaksesuaian | Berhasil |
| Ekspor hasil pemeriksaan ketidaksesuaian | Berhasil |
| Menyimpan Proyek | Berhasil |
| Membuka Proyek | Berhasil |

### 5.4.3 Evaluasi Pengujian Non-fungsional

Pengujian non-fungsional berfokus pada aspek *usability*. Analisis hasil pengujian ini didasarkan pada hasil isian kuisioner yang berupa pernyataan-pernyataan dari kakas bantu. Nilai dari pernyataan yang telah dijelaskan pada sub-bab 5.3.3 dikelompokkan ke kategori sangat tidak setuju, tidak setuju,netral,setuju dan sangat setuju. Dari hasil kuisioner yang didapatkan pada tabel 5.16 didapati bahwa dari pernyataan “Kakas bantu dapat digunakan tanpa bantuan orang lain” mendapat

persentase yang paling rendah. Ini terjadi karena beberapa partisipan belum akrab terhadap melakukan ekstraksi diagram urutan melalui Aplikasi *star-Uml* dan ekstraksi kode sumber dengan menggunakan aplikasi *xmltranslatorapp*.

# BAB VI
# KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari pengerjaan Tugas akhir dan hasil pengujian yang telah dilakukan. Selain itu,terdapat saran yang diharapkan dapat mengembangkan hasil Tugas Akhir ini lebih baik lagi

## 6.1 Kesimpulan

Selama proses pengerjaan Tugas akhir ini di dapatkan kesimpulan sebagai berikut :

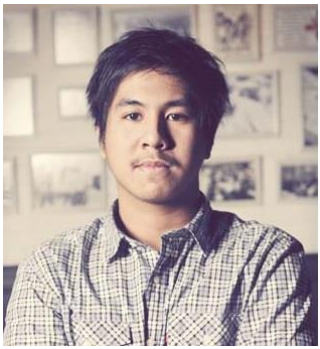
1. Kakas Bantu yang dibuat berhasil memenuhi tujuan utamanya untuk mendeteksi ketidaksesuaian kode sumber terhadap diagram urutan.
2. Berdsasarkan hasil dari kappa statistic yang didapatkan, hasil deteksi ketidaksesuaian yang dihasilkan oleh kakas bantu menghasilkan kesepakatan dengan ahli sejumlah 0.95. nilai ini menunjukkan kesepakatan antara kakas bantu dan ahli rekayasa perangkat lunak yang termasuk dalam *almost perfect*
3. Berdasarkan evaluasi non-fungsional dari hasil uji coba yang dilakukan partisipan menyetujui bahwa kakas bantu ini mudah digunakan, tetapi agak kurang mudah apabila tidak didampingi dengan orang lain
4. Dari pengerjaan kakas bantu didapat ekstraksi diagram urutan dilakukan dengan menggunakan SAX sebagai pembaca xmi dan pengambilan elemennya menggunakan metode Flag,dimana menandai setiap elemen yang ditemukan.Sedangkan untuk kode sumber didapati ekstraksi diagram urutan menggunakan SAX sebagai pembaca xml dan pengambilan elemennya menggunakan metode stack dimana setiap elemen tag xml yang ditemukan disimpan ke dalam stack.

## 6.2 Saran

Berikut adalah saran untuk pengembangan kakas bantu untuk masa yang akan datang :

1. Kakas bantu dapat mendeteksi kode sumber yang lebih kompleks lagi
2. Kakas bantu dapat mendeteksi lebih akurat dalam mendeteksi kasus diagram urutan terhadap kode sumber yang kompleks
3. Kakas bantu dapat membedakan antara jenis *message* seperti *reply* yang terdapat pada diagram urutan
4. Kakas bantu dapat mengekstraksi kode sumber lebih baik lagi
5. Kakas bantu dapat menampilkan hasil ke dalam gambaran yang lebih baik lagi

## BIODATA PENULIS



Albert Bungaran Manik lahir pada 08 Desember 1995 di Kota Surabaya. Penulis menempuh pendidikan SD di SD Santo Yusuf tropodo (2001-2007), SMP di SMP Santo Yosef Joyoboyo (2007-2010), SMA di SMA Petra 5 (2010-2013), dan S1 di Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya (2013-2018).Selama kuliah di Teknik Informatika ITS, penulis pernah menjadi BPH Schematics dan anggota organisasi mahasiswa di ITS . Pada pengerjaan tugas akhir, penulis mengambil bidang Rekayasa Perangkat Lunak (RPL) Penulis dapat dihubungi melalui alamat email **mebungaranmanik@gmail.com**

***(Halaman ini sengaja dikosongkan)***

## DAFTAR PUSTAKA


[1] "XML Translator." *PHP to XML Converter*, xmltranslator.appspot.com/sourcecodetoxml.html.

[2] Landis, J.R. and Koch,G. G. 1997 "The meassurement of observer agreement for categorical data"*biometrics*,pp.159-174.

[3] Siahaan D. 2012 *Analisa Kebutuhan dalam Rekayasa Perangkat Lunak*. Yogyakarta: Penerbit ANDI.

[4] Gwet, K. 2002. "Kappa statistic is not satisfactory for assessing the extent of agreement between raters". *statistical methods for interater reliability assessment*,No. 1, Vol. 6,pp. 1-6

[5] MKLab. 2014. *starUML*. <URL:http://staruml.io>

[6] iText Software Corp. 2000. *Core java library + PDF/A,xtra and xml worker* <URL: https://itextpdf.com/>

[7] Duric, Z., and D. Gasevic. "A Source Code Similarity System for Plagiarism Detection." *The Computer Journal*, vol. 56, no. 1, 2012, pp. 70–86.

***(Halaman ini sengaja dikosongkan)***

# LAMPIRAN A
# LANJUTAN KELAS ANALISIS

Lampiran ini merupakan deskripsi dan penjelasan kelas analisis lanjutan dari sub bab 3.1.9

## A.1 Kelas Analisis membuka proyek

Kasus penggunaan (*use case*) "membuka proyek" merupakan ekstensi dari kasus penggunaan "Memeriksa ketidaksesuaian diagram urutan". saat analis sistem telah membuka kakas bantu analis memilih pilihan "membuka proyek" pada menubar kakas bantu. Analisis kasus penggunaan ini dilakukan melalui langkah-langkah sebagai berikut :

**2. Melengkapi penjelasan kasus penggunaan (*Use case*)**

Kasus penggunaan ini dimulai saat analis sistem telah membuka kakas bantu. Analis sistem kemudian memilih menu membuka proyek pada menu bar pada kakas bantu. Lalu sistem menampilkan jendela pemilihan file. Kemudian analis sistem memilih file project yang sudah disimpan sebelumnya. Kemudian sistem mengolah file project yang sudah dibuka tadi dan menampilkan project yang sudah di save sebelumnya.

**2. Untuk setiap realisasi kasus penggunaan (*Use case*)**

**2.1 Menemukan kandidat kelas dari perilaku kasus penggunaan (*Use case*)**

Alur kasus penggunaan diatas kemudian dianalisis perilakunya untuk mendapatkan kelas. Pendekatan yang dilakukan sama dengan pendekatan yang digunakan pada kasus penggunaan sebelumnya. Kandidat kelas dapat ditemukan melalui pemilihan kata benda. Tabel 3.5 menunjukkan kandidat kelas hasil analisis melalui pendekatan kata benda.

| No. | Kandidat | Pengertian | Kelas | Penjelasan |
|---|---|---|---|---|
| **1** | Menu Membuka proyek | Menu yang digunakan untuk membuka proyek yang sudah ada di dalam kakas bantu | Bukan | Menu ini merupakan salah satu komponen penyusun layar utama kakas bantu |
| **2** | Jendela Pemilihan file | Jendela yang digunakan untuk memilih File Proyek yang sudah disimpan sebelumnya | Ya | Jendela ini merupakan antarmuka yang menangani pemilihan lokasi penyimpanan laporan |
| **3** | Membuka proyek | Proses untuk membuka file proyek yang sudah disimpan sebelumnya. | Ya | Proses ini nantinya membuka file proyek yang sudah disimpan sebelumnya. Bentuk dari proyek ini berbentuk .pkdu |
| **4** | Layar utama kakas bantu | Merupakan antarmuka utama kakas bantu yang menghubungkan analis sistem dengan kakas bantu | Ya | Layar kakas bantu terdiri dari komponen-komponen interface seperti opsi menubar, tombol. |

## 2.2 Membagi perilaku kasus penggunaan (*Use case*) ke kelas analisis

Kandidat-kandidat kelas yang telah terpilih kemudian dijabarkan ke dalam stereotype kelas. Stereotype tersebut antara lain kelas

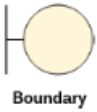
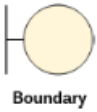
*boundary*, kelas *control*, dan kelas *entity*.Tidak jauh beda dengan penjelasan kelas analasis dengan yang sebelumnya. Kelas boundary merupakan kelas yang menghubungkan entitas luar sistem dengan sistem. Dengan kata lain kelas ini berupa form,window,sistem, atau interface. Kelas control adalah kelas yang mengendalikan aliran logika sebuah kasus penggunaan dan mengoordinasikan objek lain. Kelas entity adalah kelas yang menyimpan informasi atau data yang digunakan oleh sistem.berdasarkan tabel 3.6 pemetaan kandidat kelas yang terpilih ke dalam nama kelas dan jenis kelas pada kakas bantu dijelaskan secara detail sebagai berikut

| **Kandidat** | **Nama** | **Deskripsi** | **Jenis** |
|---|---|---|---|
| Layar utama kakas bantu | *MainPage* | Merepresentasikan objek antarmuka dari kakas bantu untuk melakukan pemilihan file yang sudah disimpan sebelumnya |  Boundary |
| Membuka Proyek | ResourceManager | Merepresentasikan objek yang mengatur proses pembukaan berkas yang sudah disimpan Sebelumnya |  Controller |
| Jendela pemilihan file | FileChooser | Merepresentasikan objek antarmuka untuk melakukan pemilihan berkas yag sudah disimpan pada proses sebelumnya |  Controller |

**3. Untuk setiap hasil kelas analisis**

**3.1 Mendeskripsikan tanggung jawab kelas**

Tidak jauh berbeda dengan penjelasan kelas analisis kasus penggunaan sebelumnya, hasil kelas analisis yang telah dipetakan sebelumnya kemudian dideskripsikan tanggungjawabnya. Tanggung jawab kelas mendeskripsikan apa yang dilakukan oleh objek tersebut untuk memenuhi tanggung jawabnya. Dari tanggung jawab yang dilakukan, hasil apa yang didapatkan oleh objek tersebut. Tanggung jawab kelas dapat diidentifikasi melalui kata kerja yang ada pada alur kasus penggunaan (*use case*). Hasil identifikasi kata kerja dijabarkan oleh tabel berikut.

| Nama | Notasi | Tanggung Jawab |
|---|---|---|
| *MainPage* |  | Membuka proyek pendeteksian yang sudah disimpan sebelumnya (*openProjectFile*) |
| *ResourceManager* |  | Memproses berkas proyek pendeteksian yang sudah disimpan sebelumnya *(load)* |
| *FileChooser* |  | Menampilkan jendela pemilihan berkas untuk memilih berkas yang sudah disimpan sebelumnya. |

**3.2 Mendeskripsikan atribut dan hubungan kelas**

Kelas-kelas yang telah dijelaskan tanggung jawabnya, dijabarkan kembali ke dalam atribut hubungannya dengan kelas lain. Atribut digunakan untuk menyimpan data atau informasi sebuah kelas, sedangkan hubungan kelas menunjukkan komunikasi antar kelas-

kelas yang memilikki tujuan untuk menjalankan kasus penggunaan (*use case*). Penjabaran atribut ditunjukkan oleh tabel berikut

| **Nama** | **Atribut** | **Hubungan** |
|---|---|---|
| *MainPage* | Opsi menubar "membuka proyek","close proyek",dan "membuat proyek" | FileChooser |
| *FileChooser* | | *ResourceManager* |
| *ResourceManager* | | *MenuBar* |

## A.2 Kelas analisis membuat proyek

Kasus penggunaan (*use case*) "membuat proyek" memilikki relasi dengan notasi *include* dengan kasus penggunaan (*use case)* "memeriksa ketidaksesuaian diagram urutan". saat analis sistem sudah membuka kakas bantu,sistem menampilkan halaman awal kakas bantu. Lalu analis sistem harus melakukan "membuat proyek" terlebih dahulu untuk bisa melakukan kasus penggunaan "memeriksa ketidaksesuaian". Analisis kasus penggunaan ini dilakukan melalui langkah-langkah sebagai berikut:

### 1. Melengkapi penjelasan kasus penggunaan (*Use case*)

Kasus penggunaan ini dimulai saat analis sistem sudah membuka kakas bantu. Analis memilikki pilihan untuk membuka proyek atau membuat proyek. Untuk bisa melakukan kasus penggunaan "mendeteksi ketidaksesuaian" maka analis sistem harus membuat proyek terlebih dahulu, dengan menekan pilihan menu "membuat proyek" pada menubar kakas bantu. Setelah menekan menubar sistem menampilkan halaman dari kasus penggunaan "mendeteksi ketidaksesuaian".

**2. Untuk setiap realisasi kasus penggunaan (*Use case*)**

**2.1 Menemukan kandidat kelas dari perilaku kasus penggunaan (*Use case*)**

Alur kasus penggunaan diatas kemudian dianalisis perilakunya untuk mendapatkan kelas. Pendekatan yang dilakukan sama dengan pendekatan yang dilakukan sebelumnya. untuk kandidat kelas yang telah diseleksi dapat dilihat pada tabel berikut.

| No | Kandidat | Pengertian | Kelas ? | Penjelasan |
|---|---|---|---|---|
| **1** | Menu membuat proyek | Salah satu komponen menubar dari object antar muka, yang berfungsi untuk membuat proyek pendeteksian yang baru | Bukan | Menu membuat proyek merupakan komponen halaman awal dari kakas bantu |
| **2** | Halaman awal kakas bantu | Antarmuka awal kakas bantu yang ditunjukkan pada saat pertama kali analis sistem mebuka kakas bantu | Ya | Merupakan antarmuka yang merupakan penghubung analis dengan kakas bantu |

**2.2 Membagi perilaku kasus penggunaan (*Use case*) ke kelas analisis**

Kandidat-kandidat kelas yang sudah seleksi kemudian dijabarkan ke tiga stereotype kelas. Stereotype kelas itu meliputi *boundary* yang merupakan penghubung entitas luar sistem dengan sistem,

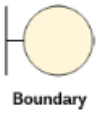
lalu *controller* yang merupakan kelas yang mengontrol aliran logika pada sistem, dan *entity* yang menjadi tempat penyimpanan data atau informasi suatu sistem. Kandidat kelas analisis dijelaskan pada tabel berikut.

| Kandidat | Nama | Deskripsi | Jenis |
|---|---|---|---|
| Halaman awal kakas bantu | *MainPage* | Merepresentasikan salah satu objek antarmuka dari kakas bantu yang meliputi menubar |  |
| - | *FileController* | Merepresentasikan objek yang memberikan instruksi untuk membuat proyek pada antar muka kakas bantu |  |

## 3. Untuk setiap hasil kelas analisis

### 3.1 Mendeskripsikan tanggung jawab kelas

Tidak jauh berbeda dengan penjelasan kelas analisis penggunaan sebelumnya , hasil analisis kelas yang sudah diseleksi sebelumnya dideskripsikan tanggung jawabnya. Tanggung jawab kelas mendeskripsikan apa yang harus dilakukan oleh objek tersebut untuk memenuhi tanggung jawabnya. Untuk setiap hasil analisis dan beserta deskripsinya dijabarkan oleh tabel berikut.

| Nama | Notasi | Tanggung Jawab |
|---|---|---|
| *MainPage* |  | Mengirimkan perintah untuk menampilkan komponen-komponen dari halaman kakas bantu |

| | | |
|---|---|---|
| *FileController* |  | Mengirimkan instruksi kepada kakas bantu untuk membuat proyek pendeteksian diagram urutan yang baru |

## 3.2 Mendeskripsikan atribut dan hubungan kelas

Kelas-kelas yang telah dijabarkan diatas telah menjelaskan tanggung jawab yang dimilikki oleh masing-masing kelas. Kelas–kelas tersebut nantinya dijabarkan kembali ke dalam atribut dan hubungannya dengan kelas lain. Atribut digunakan untuk menyimpan data atau informasi dari sebuah kelas, sedangkan hubungan kelas menunjukkan komunikasi antar kelas-kelas yang memilikki tujuan untuk menjalankan kasus penggunaan (*use case*). Penjabaran dari atribut dan hubungan kelas ditunjukkan pada tabel berikut.

| **Nama** | **Atribut** | **Hubungan** |
|---|---|---|
| *MainPage* | menu pada “menu bar” | FileController |
| *FileController* | | - |

## A.3 Kelas analisis menyimpan proyek

Kasus penggunaan (*use case*) “menyimpan proyek” merupakan ekstensi dari kasus penggunaan “Memeriksa ketidaksesuaian diagram urutan”. saat analis sistem telah membuka kakas bantu dan sudah didalam halaman kasus penggunaan “memeriksa ketidaksesuaian” analis sistem diberikan pilihan untuk melakukan kasus penggunaan “menyimpan proyek”. Maka dari itu relasi dari kasus penggunaan ini adalah extend.

**1. Melengkapi penjelasan kasus penggunaan**

Kasus penggunaan ini dimulai pada saat analis sistem saat sudah masuk kasus penggunaan ini, sistem menampilkan halaman utama kakas bantu yang berisi tombol-tombol yang mendukung halaman

utama kakas bantu. Lalu selanjutnya setelah analis sistem sudah melakukan pendeteksian, analis sistem menekan tombol simpan. Lalu sistem menampilkan jendela pemilihan tempat penyimpanan file, lalu analis menekan tombol save. Dan kakas bantu kemudian menyimpan proyek pendeteksian dengan format ".pkdu" pada tempat penyimpanan yang sudah di pilih oleh analis sistem

**2. Untuk setiap realisasi kasus penggunaan (*Use case*)**

**2.1 Menemukan kandidat kelas dari perilaku kasus penggunaan (*Use case*)**

Alur kasus penggunaan diatas kemudian di analisis perilakunya untuk mendapatkan kelas. Pendekatan yang dilakukan sama dengan pendekatan yang dilakukan pada kelas analisis sebelumnya. kandidat kelas bisa ditemukan melalui pilihan kata. Untuk kandidat kelas pada kelas analisis ini dapat dilihat pada tabel berikut ini

| No | Kandidat | Pengertian | Kelas? | Penjelasan |
|---|---|---|---|---|
| **1** | Analis sistem | Peran yang memilikki tanggung jawab untuk pengembangan dalam sebuah Perangkat lunak | Bukan | Analis sistem merupakan pengguna dari kakas bantu ini. |
| 2 | Halaman utama kakas bantu | Antar muka utama kakas bantu yang mengghubungkan analis sistem dengan kakas bantu | Ya | Halaman utama kakas bantu terdiri dari komponen seperti Tombol untuk menyimpan, tombol untuk memproses, dan teks area hasil pendeteksian |
| 3 | Tombol simpan | Merupakan salah satu elemen dari kakas | Bukan | Tombol simpan merupakan salah |

| | | | | |
|---|---|---|---|---|
| | | bantu yang berguna untuk memilih lokasi penyimpanan | | satu komponen dari halaman utama kakas bantu |
| 4 | Jendela pemilihan tempat penyimpanan file | Jendela untuk memilih lokasi penyimpanan dimana berkas proyek akan disimpan. Proyek akan disimpan dengan ekstensi ".pkdu" | Ya | Jendela ini merupakan antarmuka yang menangani pemilihan lokasi dimana berkas proyek akan disimpan, antar muka ini memilikki komponen seperti tombol "save" |
| 5 | Menyimpan proyek pendeteksian | Proses untuk melakukan penyimpanan dari hasil proses pendeteksian, penyimpanan dilakukan dengan menyimpan hasil ekstraksi diagram urutan, hasil ekstraksi kode sumber, dan hasil deteksi ketidak sesuaian | Ya | Proses ini mengatur untuk obyek yang akan disimpan oleh kakas bantu dengan menseleksi obyek yang akan disimpan. |

## 2.2 Membagi perilaku kasus penggunaan (*Use case*) ke kelas analisis

Kandidat-kandidat kelas yang sudah seleksi kemudian dijabarkan ke tiga stereotype kelas. Stereotype kelas itu meliputi *boundary* yang merupakan penghubung entitas luar sistem dengan sistem,

lalu *controller* yang merupakan kelas yang mengontrol aliran logika pada sistem, dan *entity* yang menjadi tempat penyimpanan data atau informasi suatu sistem. kelas analisis dijelaskan pada tabel berikut.

| Kandidat | Nama | Deskripsi | Jenis |
|---|---|---|---|
| Halaman utama kakas bantu | *MainPage* | Merepresentasikan objek antarmuka dari kakas bantu . |   |
| Jendela pemilihan tempat penyimpanan file | *FileChooser* | Merepresentasikan objek antarmuka untuk memilih lokasi penyimpanan file proyek ketidaksesuaian |   |
| Menyimpan proyek pendeteksian | *ResourceManager* | Merepresentasikan objek yang mengatur logika untuk penyimpanan berkas yang akan disimpan |   |
| - | *FileController* | Merepresentasikan objek yang mengatur komunikasi antar antar muka dengan resource manager |   |

**3. Untuk setiap hasil analisis**

**3.1 Mendeskripsikan tanggung jawab kelas**

Tidak jauh berbeda dengan penjelasan kelas analisis penggunaan sebelumnya , hasil analisis kelas yang sudah diseleksi sebelumnya dideskripsikan tanggung jawabnya. Tanggung jawab kelas mendeskripsikan apa yang harus dilakukan oleh objek tersebut

untuk memenuhi tanggung jawabnya.Untuk setiap hasil analisis dan beserta deskripsinya dijabarkan oleh tabel berikut

| Nama | Notasi | Tanggung Jawab |
|---|---|---|
| *MainPage* |  | Mengirimkan perintah untuk menampilkan halaman antar muka dari kakas bantu |
| *FileChooser* |  | Menampilkan jendela pemilihan berkas yang akan dipilih oleh analis sistem |
| *ResourceManager* |  | Mengirimkan perintah untuk melakukan penyimpanan untuk kakas bantu (*save)* |
| *FileController* |  | Mengirimkan berkas yang sudah dipilih oleh analis melalui *FileChooser* sebelumnya |

## 3.2 Mendeskripsikan atribut dan hubungan kelas

Kelas-kelas yang telah dijabarkan diatas telah menjelaskan tanggung jawab yang dimilikki oleh masing-masing kelas. Kelas–kelas tersebut nantinya dijabarkan kembali ke dalam atribut dan hubungannya dengan kelas lain. Atribut digunakan untuk menyimpan data atau informasi dari sebuah kelas, sedangkan hubungan kelas menunjukkan komunikasi antar kelas-kelas yang memilikki tujuan untuk menjalankan kasus penggunaan (*use case*). Penjabaran dari atribut dan hubungan kelas ditunjukkan pada tabel berikut.

| Nama | Attribut | Hubungan |
|---|---|---|
| *MainPage* | Tombol “simpan” | *FileChooser* |
| *FileChooser* | - | *FileController* |
| *FileController* | | *ResourceManager* |
| *ResourceManager* | | - |

## A.4 Kelas analisis mengekspor hasil laporan

Kasus penggunaan (*use case*) “Mengekspor hasil laporan” merupakan ekstensi dari kasus penggunaan “Memeriksa ketidaksesuaian diagram urutan”. saat analis sistem telah membuka kakas bantu dan sudah didalam halaman kasus penggunaan “memeriksa ketidaksesuaian” analis sistem diberikan pilihan untuk melakukan kasus penggunaan “Mengekspor hasil laporan”. Maka dari itu relasi dari kasus penggunaan ini adalah extend.

**1. Melengkapi penjelasan kasus penggunaan (*Use case*)**

Kasus penggunaan ini dimulai saat analis sistem telah melakukan pendeteksian diagram urutan. Analis sistem memilikki dua pilihan apakah analis sistem ingin mengekspor hasil pendeteksian atau tidak. Apabila analis sistem ingin mengekspor hasil pendeteksian maka analis sistem menekan tombol “ekspor ke pdf” pada halam utama pendeteksian. Setelah analis sistem menekan tombol tersebut sistem menampilkan jendela pemilihan berkas. Setelah itu analis sistem memilih lokasi dimana hasil ekspor pdf akan disimpan. Setelah analis memilih lokasi, analis sistem menekan tombol save. Lalu sistem mengekspor hasil analisis, dan menyimpannya di lokasi yang telah dipilih oleh analis sistem

**2. Untuk setiap realisasi kasus penggunaan (*Use case*)**

**2.1 Menemukan kandidat kelas dari perilaku kasus penggunaan (*Use case*)**

Alur kasus penggunaan diatas kemudian dianalisis perilakunya untuk mendapatkan kelas. Pendekatan yang dilakukan untuk

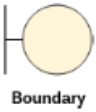
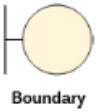
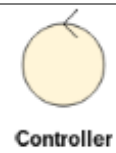
mendapatkan kelas, sama dengan pendekatan dari kelas analisis sebelumnya. Kandidat kelas bisa ditemukan melalui tabel berikut

| No | Kandidat | Pengertian | Kelas? | Penjelasan |
|---|---|---|---|---|
| **1** | Halaman utama layar pendeteksian | Peran yang memilikki tanggung jawab untuk menjembatani analis sistem dengan kakas bantu | Ya | Halaman utama kakas bantu terdiri dari komponen-komponen seperti tombol,teks area yang menyusun halaman utama layar pendeteksian |
| **2** | Tombol “ekspor ke pdf” | Merupakan salah satu komponen dari antarmuka pendeteksian | Bukan | Tombol ini merupakan salah satu komponen penyusun dari halaman utama layar pendeteksian |
| **3** | Jendela pemilihan lokasi penyimpanan | Merupakan jendela untuk menentukan penyimpanan lokasi dimana PDF akan disimpan | Ya | Jendela ini merupakan antar muka yang menangani pemilihan lokasi penyimpanan hasil ekspor pdf dari kakas bantu |
| **4** | Mengekspor hasil analisis | Merupakan proses untuk melakukan ekspor dari hasil deteksi yang sudah dikeluarkan oleh kakas bantu ke | Ya | Proses ini mengatur proses dari ekspor yang didapatkan dari kakas bantu menjadi sebuah berkas dengan |

| | | | |
|---|---|---|---|
| | dalam “.pdf” | ekstensi | berekstensi “.pdf” |

## 2.2 Membagi perilaku kasus penggunaan (*Use case*) ke kelas analisis

Kandidat-kandidat kelas yang sudah seleksi kemudian dijabarkan ke tiga stereotype kelas. Stereotype kelas itu meliputi *boundary* yang merupakan penghubung entitas luar sistem dengan sistem, lalu *controller* yang merupakan kelas yang mengontrol aliran logika pada sistem, dan *entity* yang menjadi tempat penyimpanan data atau informasi suatu sistem. kelas analisis dijelaskan pada tabel 3.18

| **Kandidat** | **Nama** | **Deskripsi** | **Jenis** |
|---|---|---|---|
| Halaman utama layar pendeteksian | *MainPage* | Merepresentasikan objek antarmuka utama dari kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan kode sumber |  |
| Jendela pemilihan | *FileChooser* | Merepresentasikan objek antarmuka untuk memilih lokasi penyimpanan laporan “.pdf” |  |
| Mengekspor hasil analisis | *PDFwriter* | Merepresentasikan objek yang mengatur proses untuk merubah hasil deteksi menjadi berkas dengan ekstensi “.pdf” |  |

**3. Untuk setiap hasil kelas analisis :**

**3.1 Mendeskripsikan tanggung jawab kelas**

Tidak jauh berbeda dengan penjelasan kelas analisis penggunaan sebelumnya , hasil analisis kelas yang sudah diseleksi sebelumnya dideskripsikan tanggung jawabnya. Tanggung jawab kelas mendeskripsikan apa yang harus dilakukan oleh objek tersebut untuk memenuhi tanggung jawabnya. Untuk setiap hasil analisis dan beserta deskripsinya dijabarkan oleh tabel 3.19

| Nama | Notasi | Tanggung jawab |
|---|---|---|
| *MainPage* |  | Menyediakan komponen-komponen antarmuka untuk melakukan proses ekspor ke pdf, dalam kasus ini komponen yang dimaksud adalah tombol "ekspor ke pdf" |
| *FileChooser* |  | Memilih lokasi penyimpanan hasil ekspor pdf yang dilakukan oleh analis sistem |
| *PDFwriter* |  | Melakukan proses perubahan dari hasil yang diterima dari kakas bantu menjadi berkas dokumen berbentuk ".pdf" |

**3.2 Mendeskripsikan atribut dan hubungan kelas**

Penjabaran hubungan dari atribut dan hubungan kelas dijelaskan pada tabel berikut

| Nama | Atribut | Hubungan |
|---|---|---|
| *MainPage* | Tombol “ekspor ke pdf” | *FileChooser* |
| *FileChooser* | | *PDFwriter* |
| *PDFwriter* | | - |

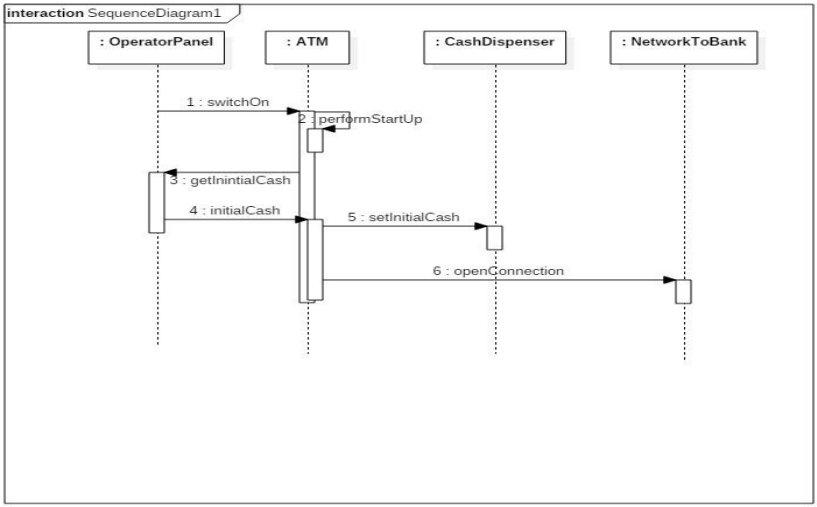
# LAMPIRAN B
# DATA PENGUJIAN

Data pengujian yang digunakan untuk menjadi dataset didapatkan dari tautan-tautan yang berisi penjelasan pada sub-bab lampiran ini. Terdapat 5 buah data uji berupa diagram urutan beserta kode sumber nya, diagram urutan diambil berupa fitur dari sebuah sistem atau sebuah contoh yang terdapat dari berbeagai tautan.

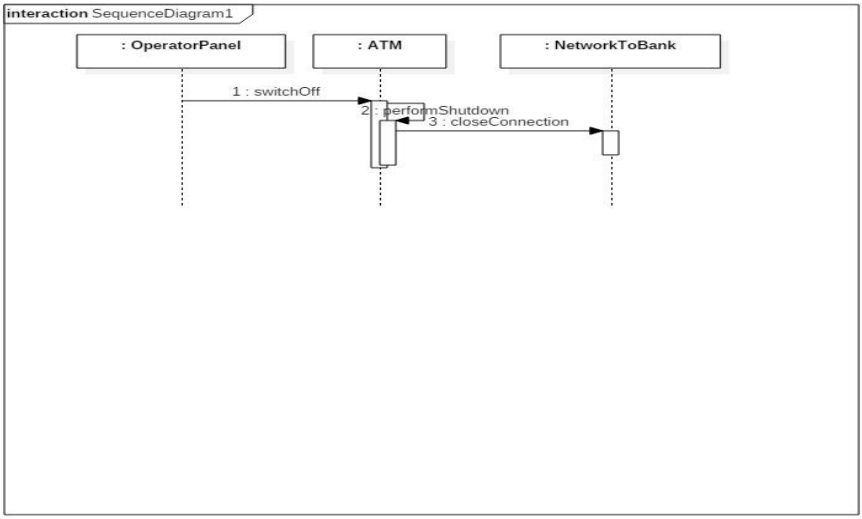
## B.1 Kasus ATM

Pada kasus dataset ini, diambil dari laman *http://www.math-cs.gordon.edu/courses/cs211/ATMExample/*. Pada kasus ini menunjukkan contoh kasus ATM yang dimana bisa menjalankan *perform startup* , *perform shutdown,transaction*. Pada kasus ini hanya beberapa kasus yang diambil. diagram urutan yang dipilih berupa :

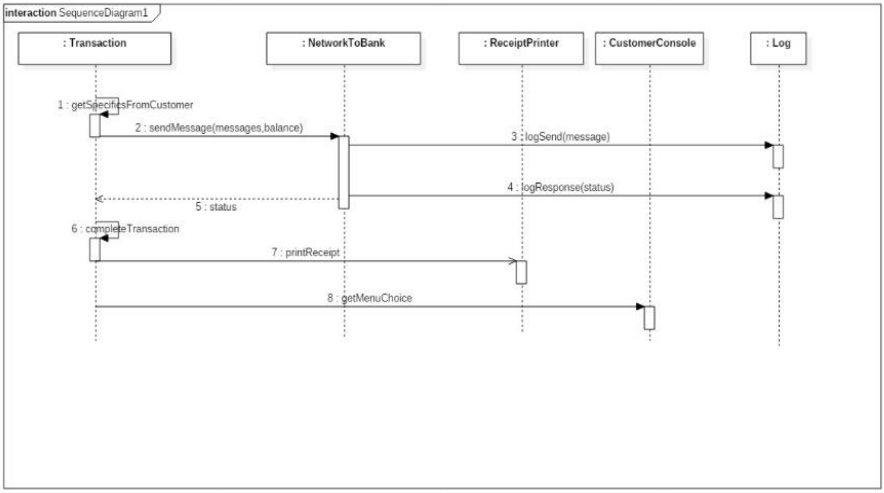
**1. Perform startup ATM**



## 2. Perform shutdown ATM



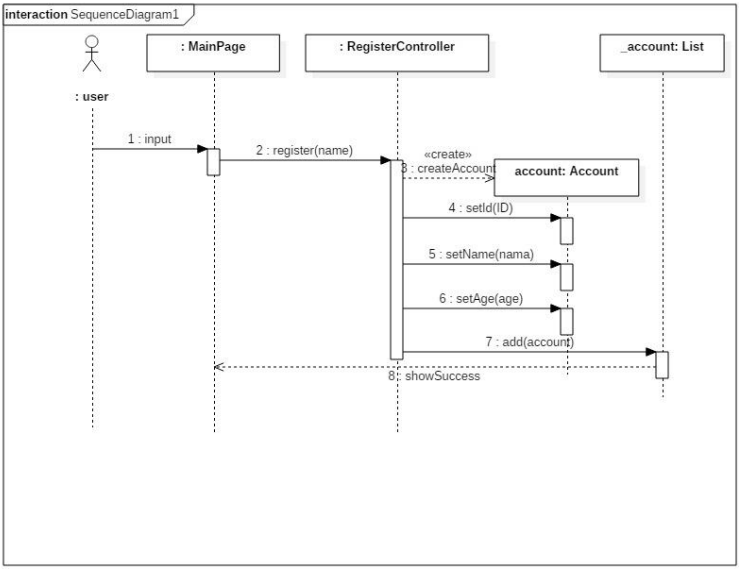

## 3. Transaction ATM



## B.2 Kasus Management Library

Pada dataset ini diambil dari laman *https://www.visual-paradigm.com/tutorials/seqrev.jsp*. Kasus ini menggambarkan simulasi dari sebuah sistem register pada kasus management library. Pada uji ini diambil satu buah diagram urutan beserta kode sumber.

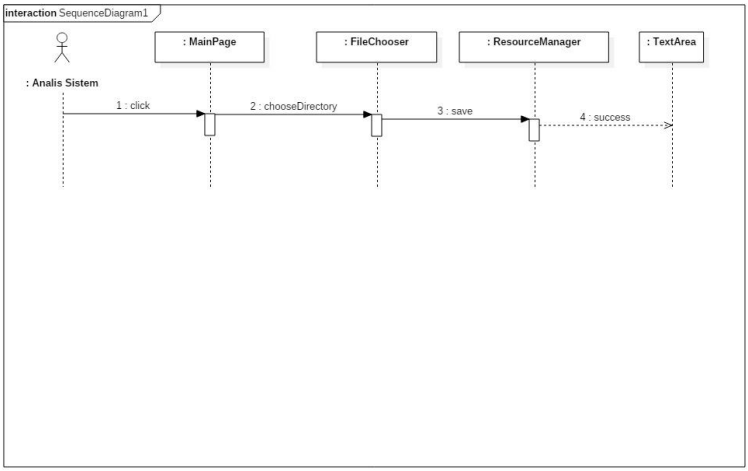
### 1. Register Management Library




## B.3 Kakas Bantu Pendeteksi ketidaksesuaian kode sumber terhadap diagram urutan

Pada dataset ini diambil dari hasil kakas bantu yang telah dibangun dari hasil proyek tugas akhir ini. Kasus ini menggambarkan fungsi save pada kakas bantu ini. Pada data uji ini diambil satu buah diagram urutan beserta kode sumber.

## 1. Fungsi save kakas bantu




Dari hasil data uji yang disediakan diatas nantinya akan didapatkan 10 buah dataset yang berisi dari tiga buah dari kasus *ATM*, satu buah dari kasus *management library*, dan satu buah dari kakas bantu ini. Dari kelima data itu nantinya akan di bandingkan antara kode sumber dari diagram urutan tersebut, dan bukan dari diagram urutan tersebut. Maka dari itu nantinya didapatkan sejumlah 10 dataset yang digunakan untuk menguji kakas bantu ini.

# LAMPIRAN C

# SYSTEM REQUIREMENT SPECIFICATION

## C.1 Pendahuluan

### C.1.1 Tujuan Penulisan Dokumen

Dokumen ini berisi Spesifikasi Kebutuhan Perangkat Lunak (SKPL) atau Software Requirement Spesification (SRS) untuk Kakas bantu ketidaksesuaian Kode sumber Terhadap Diagram Urutan. Tujuan penulisan dokumen ini adalah memberikan penjelasan mengenai hasil perangkat lunak yang akan dibangun baik berupa gambaran umum maupun penjelasan secara detil dan menyeluruh.Dokumen ini akan digunakan sebagai bahan acuan dalam proses pengembangan dan sebagai bahan evaluasi pada saat proses pengembangan perangkat lunak maupun diakhir pengembangannya. Dengan adanya dokumen SKPL ini diharapkan pengembangan perangkat lunak akan lebih terarah dan terfokus serta tidak menimbulkan ambiguitas.

### C.1.2 Lingkup Masalah

Perangkat lunak yang dibangun adalah Kakas bantu ketidaksesuaian Kode sumber Terhadap Diagram Urutan, yaitu merupakan kakas bantu untuk sistem analis mendeteksi ketidak sesuai an antara kode sumber dengan diagram urutan.kakas bantu ini berupa sebuah aplikasi desktop yang berbasis java dimana analis akan memasukkan sebuah input berkas yang berekstensi XMI dari diagram urutan dan kode sumber. Sistem dapat melakukan hal-hal sebagai berikut:

1. Memasukkan file XMI diagram urutan dan kode sumber
2. Melakukan pendeteksian ketidaksesuaian Diagram urutan dan kode sumber

3. Melakukan Export hasil pendeteksian berupa format pdf.

Dengan adanya Kakas bantu ketidaksesuaian kode sumber dengan diagram urutan,analis dapat mengetahui ketidak sesuaian kode sumber

### C.1.3 Definisi dan istilah

Berikut adalah daftar definisi dan istilah penting yang digunakan dalam dokumen SKPL ini:

- SRS :*Software Requirements Specification*, atau
- SKPL: Spesifikasi Kebutuhan Perangkat LunakDokumen hasil analisis yang berisi spesifikasi kebutuhan perangkat lunak.
- IEEE :*Institute of Electrical and Electronics Engineering* Standar internasional untuk pengembangan dan perancangan produk.

### C.1.4 Aturan Penamaan dan penomoran

Penulisan dokumen SKPL ini menggunakan berbagai macam aturan penamaan dan penomoran yang berbeda-beda untuk beberapa bagian tertentu. Aturan penamaan dan penomoran yang digunakan berdasarkan hal/bagian tersebut adalah seperti yang tercantum pada berikut ini.

| Hal/Bagian | Aturan Penomoran/Penamaan |
|---|---|
| Kebutuhan Fungsional | SKPL-FXX : Menunjukkan kebutuhan fungsional ke-XX |
| Kebutuhan Non Fungsional | SKPL-NFXX : Menunjukkan kebutuhan non fungsional ke-XX |
| Ringkasan kebutuhan fungsional | SKPL-Fxxx dimana xxx adalah tiga digit bilangan bulat dimulai dari 000 |
| Ringkasan kebutuhan non-fungsional | SKPL-NFxxx dimana xxx adalah tiga digit bilangan bulat dimulai dari 000 |

### C.1.5 Referensi

Dokumen-dokumen yang digunakan sebagai referensi dalam pembuatan SKPL ini adalah sebagai berikut :

1. Dokumen *Software Requirement Spesification (SRS) – IEEE* tahun 1999 oleh Karl E. Wiegers.
2. Panduan Penggunaan dan Pengisian Spesifikasi Perangkat Lunak (SKPL), Jurusan Teknik Informatika, Institut Teknologi Sepuluh November.
3. Panduan Penggunaan dan Pengisian Spesifikasi Perangkat Lunak (SKPL), Jurusan Teknik Informatika, Institut Teknologi Bandung.

## C.2 Deskripsi Umum Perangkat Lunak

### C.2.1 Deskripsi Umum Sistem

Kakas bantu ketidak sesuaian kode sumber dengan diagram urutan adalah kakas bantu yang digunakan sebagai sarana untuk sistem analis agar dapat mendeteksi ketidaksesuaian di dalam kode sumber terhadap diagram urutan, sehingga memudahkan sistem analis untuk menganilisis kode sumber. Didalam kakas bantu ini, input diagram urutan dan kode sumber akan berupa file XMI.

Kakas bantu yang dibangun ini memilikki bagian utama yang berupa dari sisi sistem analis saja:

1. Analis, kakas bantu menyediakan berupa memilih berkas diagram urutan, berkas kode sumber, bisa memproses ketidaksesuaian dan bisa mengeskpor kedalam bentuk PDF

### C.2.2 Fungsi Produk

Perangkat Lunak *Kakas bantu ketidak sesuaian diagram urutan dengan kode sumber* ini mempunyai beberapa fungsi utama, antara lain

1. (SKPL-F1) Dapat menganalisis ketidaksesuaian kode sumber terhadap diagram urutan
2. (SKPL-F2) Dapat mengekspor hasil analisis menjadi PDF

### C.2.3 Karakteristik Pengguna

| No | Kategori Pengguna | Tugas | Hak Akses ke aplikasi | Kemampuan yang harus dimiliki |
|---|---|---|---|---|
| 1. | Analis | Melakukan Analisis ketidaksesuaian diagram urutan terhadap kode sumber | Bisa memilih file XMI kode sumber dan diagram urutan, dan memroses ketidaksesuaian | 1.Harus paham mengenai *Diagram urutan*<br>2.Mengerti Keahlian sistem analys<br>3.Harus mengerti Kode sumber berbahasa Java |

### C.2.4 Batasan

Pengembangan *Kakas bantu ketidaksesuaian diagram urutan dengan kode sumber* ini memiliki keterbatasan-keterbatasan yaitu sebagai berikut :

1. Bahasa yang dideteksi hanya berupa bahasa pemrograman Java , dan Sudah diesktraksi.
2. File XMI diagram urutan berupa hasil ekspor dari StarUML.
3. Ekstraksi kode sumber dengan menggunakan XMLtranslator app [1]

### C.2.5 Lingkungan Operasi

Aplikasi kakas bantu ini akan berfungi dengan spesifikasi :

Platform sistem operasi : Microsoft Windows
Versi sistem operasi : Windows 7
Bahasa : Java

Teks editor : Netbeans

## C.3 Deskripsi Umum Kebutuhan

### C.3.1 Kebutuhan Antarmuka eksternal

#### C.3.1.1 Antarmuka Pengguna

Kakas bantu ketidaksesuaian diagram urutan dengan kodesumber menggunakan antarmuka grafis (GUI). Pengguna dapat menginputkan melalui *keyboard* dan *mouse* serta digunakan dengan sistem operasi *Windows.*

#### C.3.1.2 Antarmuka Perangkat Keras

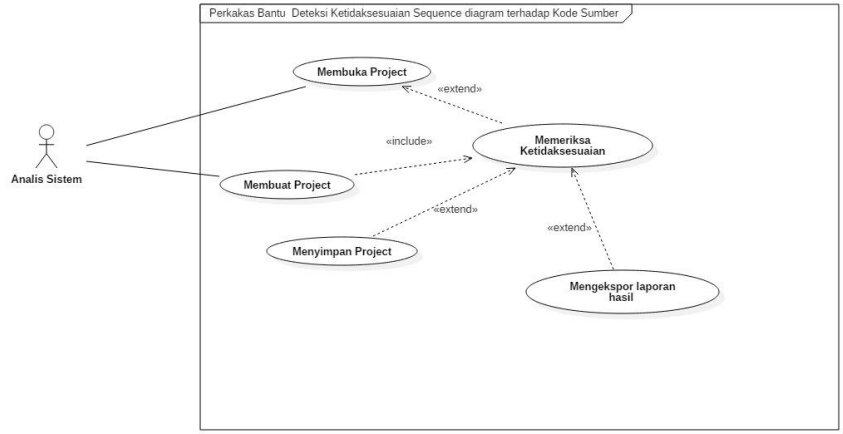
Kakas bantu ketidaksesuaian diagram urutan dengan kodesumber berjalan di computer client(sistem analis).

#### C.3.1.3 Antar Muka Komunikasi

Kekas bantu perangkat lunak akan dibangun menggunakan bahasa Java dengan GUI menggunakan pustaka java swing, dan akan berjalan di OS windows.

## C.3.2 Deskripsi Fungsional

### C.3.2.1 Use Case Diagram




### C.3.2.2 Use Case Description

**1. Memeriksa Ketidaksesuaian**

**1.1. Brief Description**

Usecase ini Mendeskripsikan bagaimana analis memeriksa ketidaksesuaian kode sumber terhadap diagram urutan.

2. Flow Of Events

2.1. Basic Flow

Sistem menampilkan halaman muka dari aplikasi

1. Analis menekan tombol Diagram Urutan untuk menambahkan diagram urutan.
2. Sistem menampilkan direktori folder
3. Analis memilih diagram urutan sesuai dengan file extensi yang sudah ditentukan.

4. Sistem menampilkan nama file diagram urutan pada jendela Daftar Diagram Urutan.
5. Analis menekan tombol Kode Sumber untuk menambahkan kode sumber.
6. Sistem menampilkan direktori Folder
7. Analis memilih direktori tempat kumpulan kode sumber dengan ekstensi .java yang hendak diproses.
8. Sistem menampilkan nama-nama file kode sumber pada jendela Daftar Kode Sumber.
9. Analis menekan tombol Proses.
10. Sistem menampilkan informasi kemajuan proses. Setelah selesai, sistem menampilkan hasil analisis.

**2.2. Alternative Flow**

Use case ini tidak memilikki alur alternatif

**2.3. Exceptional Flow**

Use case ini tidak memilikki alur eksespsional

**3. Special requirement**

Use case ini tidak memilikki special requirement

**4. Pre-Conditions**

4.1. Analis sudah membuka aplikasi kakas bantu pendeteksian ketidaksesuaian kode sumber terhadap diagram urutan

**5. Post-Conditions**

5.1. Analis mendapatkan hasil deteksi ketidaksesuaian

**6. Extension Points**

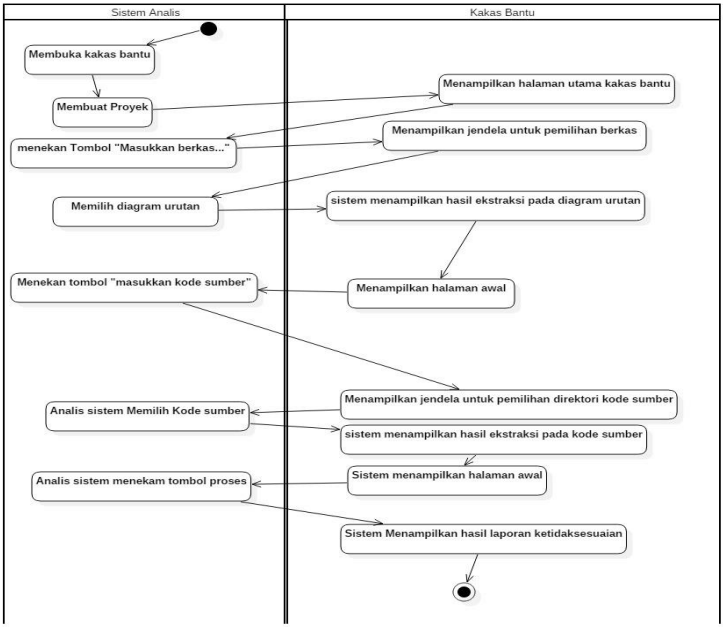
1. Analis mengekspor hasil pendeteksian ke bentuk PDF :
   1.1. Analis menekan tombol export to pdf
   1.2. Analis memilih direktori dimana hasil export akan disimpan
2. Analis Menekan tombol membuat project baru:
   2.1. Analis menekan tombol project baru

2.2. Sistem menampilkan Direktori folder.
2.3. Analis memilih direktori untuk direktori project.

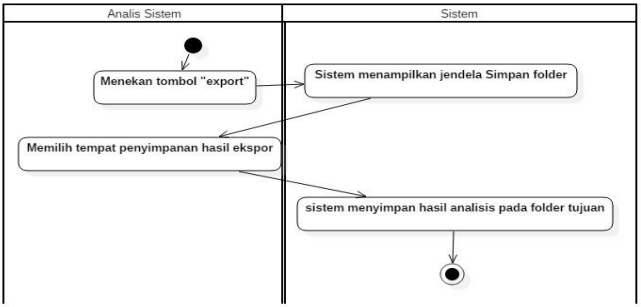
3. Analis menekan tombol menyimpan project:
   3.1. Analis menekan tombol menyimpan project
   3.2. Sistem menampilkan Alert Tanda Program telah disimpan

### C.3.2.2.1 Activity Diagrams

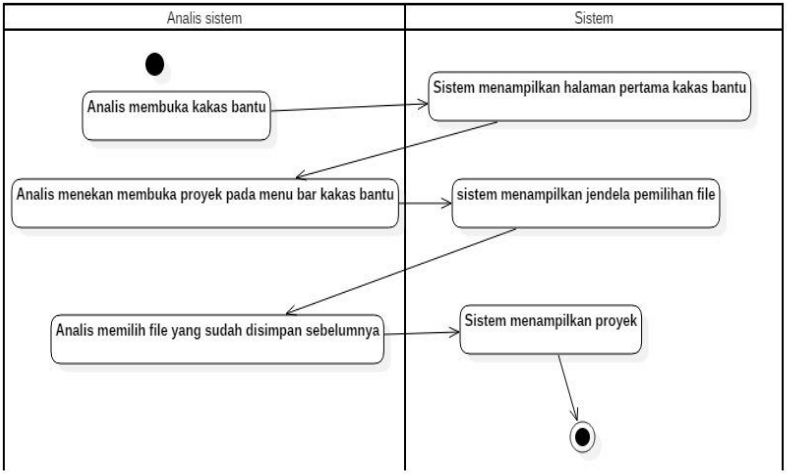
**1. Mendeteksi Ketidaksesuaian**



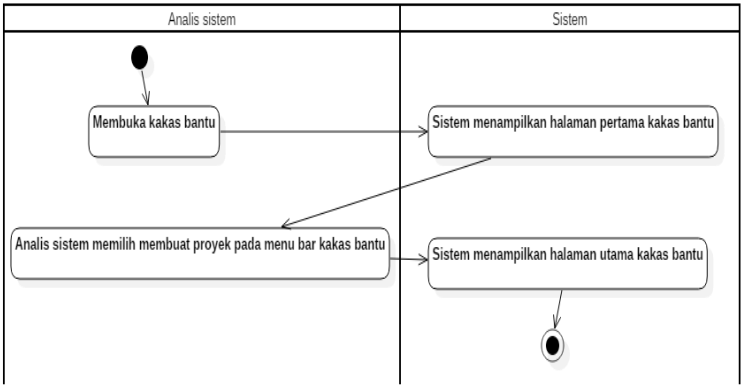

2. Mengekspor PDF




3. Membuka proyek baru

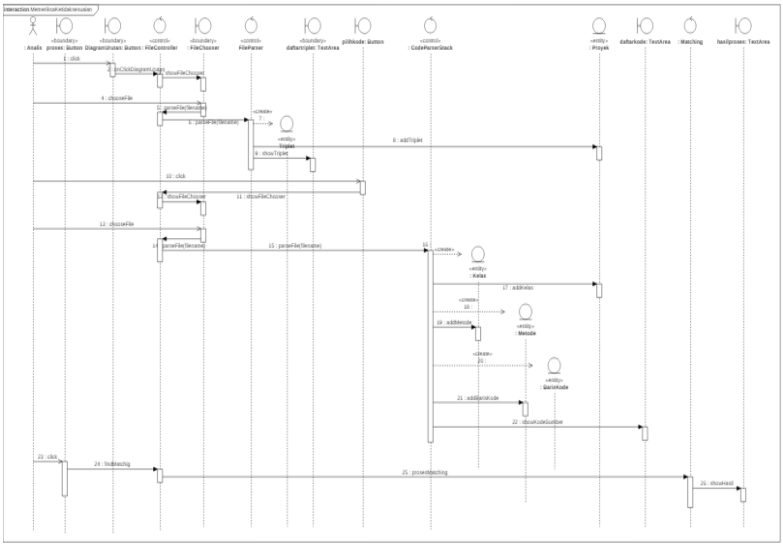



4. Membuat proyek

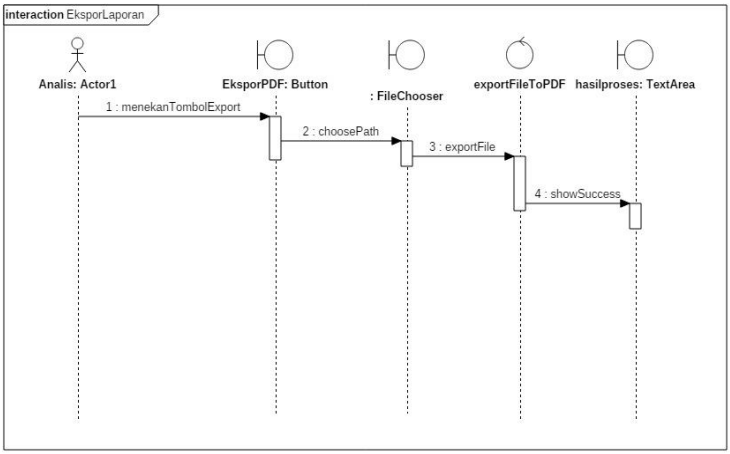



## C.3.2.2.2 Diagram Sekuens

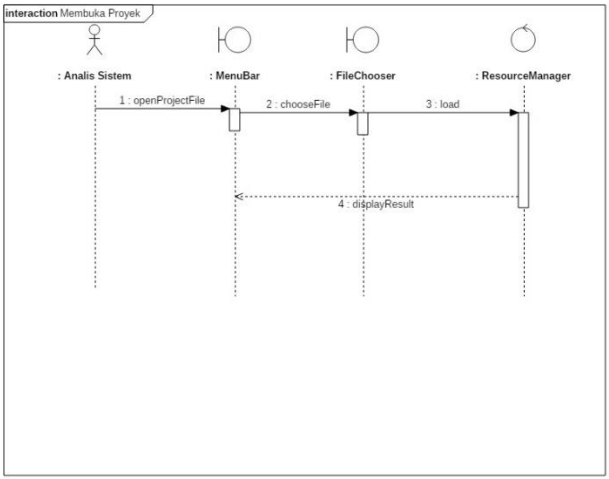
1. Mendeteksi Ketidaksesuaian



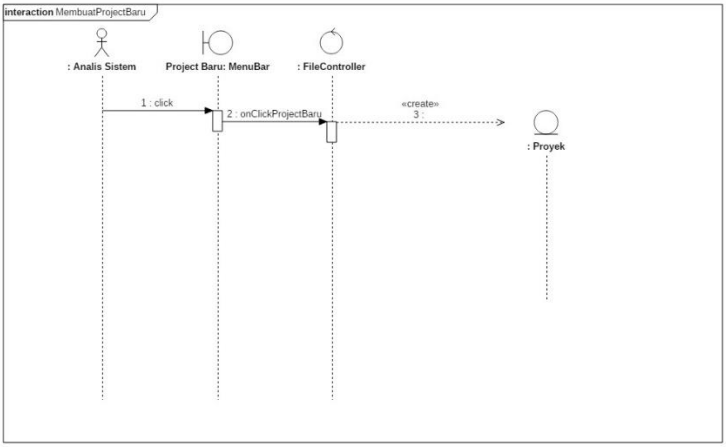
2. Mengekspor PDF



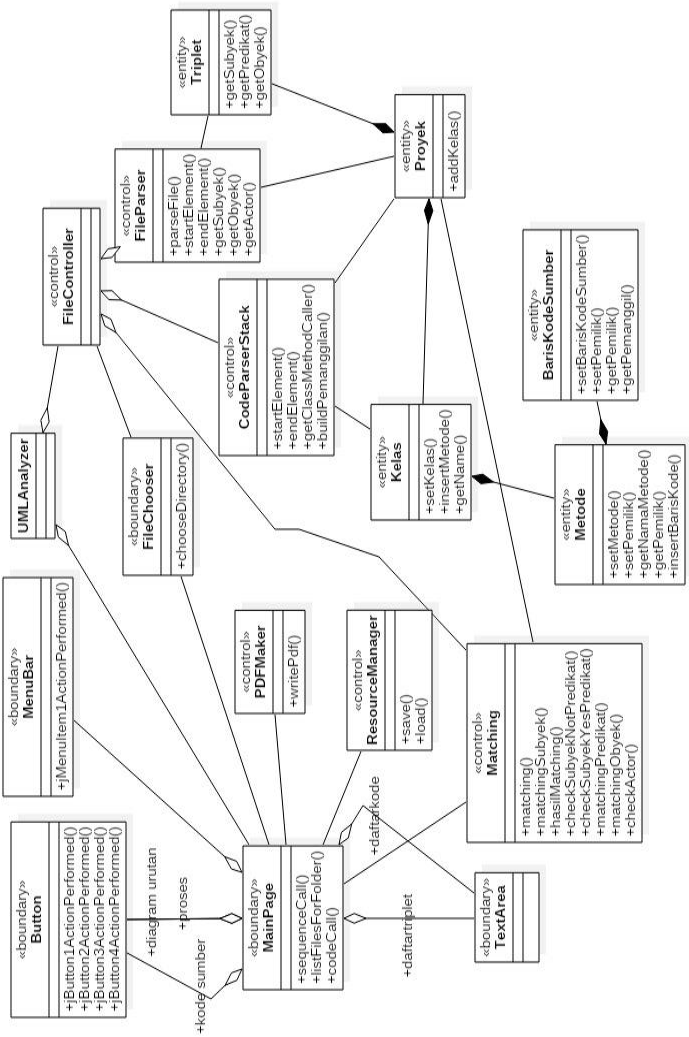

3. Membuka Proyek



4. Membuat Proyek



### C.3.3 Class Diagram



*(Halaman ini sengaja dikosongkan)*

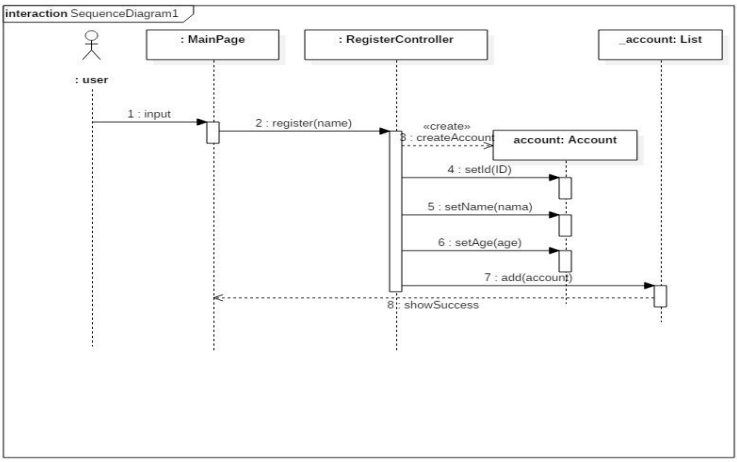
# LAMPIRAN D

# HASIL PENGUJIAN DATA KAKAS BANTU DENGAN SKENARIO BERBEDA

## D.1 Skenario word similarity

Pada bab lampiran ini,akan menunjukkan hasil dari pengujian kakas bantu dan ahli yang ditunjukkan pada kasus dimana terdapat perbedaan didalam sebuah kalimat tetapi terdapat makna yang sama dari kalimat tersebut. Dari tugas akhir ini pendekatan yang digunakan untuk menyelesaikan masalah ini adalah dengan menggunakan metode yang telah dijelaskan pada bab 3.2.2.8 yang dimana metode yang digunakan adalah word similarity yang merujuk pada word net dengan perhitungan wu-palmer.

Didapati sebuah diagram urutan dari kasus ini adalah sebuah kasus register pada management library yang ditunjukkan pada gambar dibawah ini.




Dengan kode sumber yang ditunjukkan pada dibawah ini.

```
1. public class MainPage {
2.     public void isiMainpage(String nameInput,String ageInput){
3.         RegisterController rc = new RegisterController();
4.         rc.register(nameInput,ageInput);
5.     }
6. }
```

Dari kode sumber yang telah dimasukkan ini, dilakukan proses pendeteksian ketidaksesuaian. Pertama kali kakas bantu mendeteksi user yang merupakan sebuah aktor, yang berarti alur dari aktor ini menggambarkan bagaimana entitas luar berinteraksi dengan sistem. Dalam hal ini merujuk dari kesepakatan dalam tugas akhir ini apabila subyek ataupun obyek yang merujuk pada aktor maka tidak perlu dilakukan pendeteksian sehingga langsung diberi tanda benar. Sehingga dari alur pertama dapat kita simpulkan bahwa alur pertama merupakan alur entitas luar berinteraksi dengan sistem. Sehingga alur pertama ini menghasilkan sebuah triplet dengan nilai subyek benar, predikat salah dikarenakan input bukanlah sebuah metode namun sebuah gambaran saja, dan obyek menghasilkan benar dikarenakan terdapat kelas MainPage didalam *folder* kode sumber yang telah dimasukkan

```
1.  public class RegisterController {
2.
3.    private List _accounts = new ArrayList();
4.
5.      public void entryAccount(String name, int age) {
6.          Account account = new Account();
7.          account.setId(1);
8.          account.setName(name);
9.          account.setAge(age);
10.         _accounts.add(account);
11.     }
12.     public void register(String name){
13.
14.
15.     }
16.     public List getAccounts(){
17.         return _accounts;
18.     }
19. }
```

Setelah berhasil mendeteksi alur pertama, dilanjutkan dengan alur

ke dua dimana didapati sebuah kelas *MainPage* melakukan pemanggilan metode *register* pada *RegisterController*. Didapati terdapat pemanggilan metode register milik registerController yang ditunjukkan pada baris ke 4 pada gambar kelas MainPage yang telah disertakan. Setelah itu dilanjutkan pada predikat dalam hal ini predikat merupakan metode register, didapati pada gambar kelas registerController terdapat metode register pada kelas registerController. Dilanjutkan dengan mendeteksi registerController apakah sebuah kelas, didapati memang benar sebuah kelas. Sehingga alur ke dua terbentuk subyek benar, predikat benar, obyek benar.

Setelah berhasil dilanjutkan pada alur ke tiga. Pendeteksian dilakukan dengan mendeteksi subyek terlebih dahulu, didapati registerController merupakan sebuah kelas pada kode sumber, namun tidak terdapat pemanggilan create account sehingga dapat diidentifikasi subyek diberi nilai salah. Selanjutnya Pada alur ketiga dalam tugas akhir ini dibatasi dengan pembacaan tipe message hanya berbentuk sebuah message bukan sebuah reply ataupun create message, sehingga sebenarnya alur ketiga ini predikatnya dapat kita berikan nilai salah. Namun dikarenakan di tugas akhir ini mengimplementasikan juga word similarity nilai ini bisa terpengaruh terhadap makna dari sebuah kata. Didapati pada diagram urutan predikat digambarkan dengan createAccount lalu predikat ini melakukan iterasi setiap metode pada setiap metode dalam suatu kelas. Pada setiap iterasi itu diberlakukan perhitungan similarity atau perhitungan kesamaan makna dari sebuah metode itu, penjelasan metode yang digunakan ini dapat dilihat pada bab 3.2.2.8 den bab 4.2.2.4. dari setiap iterasi tersebut setiap hasilnya akan diseleksi dengan sebuah *tresshold* yang telah disepakati sebelumnya yang menyentuh angka 0.35. Dari setiap iterasi yang

dilakukan ditemukan sebuah metode yang mendapati sebuah tressholde yang melebihi 0.35 metode itu adalah metode setName pada kelas account yang ditunjukkan pada dibawah ini.

```
1.  public class Account {
2.
3.    private int _id;
4.    private String _name;
5.    private int _age;
6.
7.    public void setId(int id){
8.      _id = id;
9.    }
10.
11.   public int getId(){
12.     return _id;
13.   }
14.
15.   public void setName(String name){
16.     _name = name;
17.   }
18.
19.   public String getName(){
20.     return _name;
21.   }
22.
23.   public void setAge(int age){
24.     _age = age;
25.   }
26.
27.   public int getAge(){
28.     return _age;
29.   }
30.
31. }
```

Untuk nilai dan fungsi perhitungannya dapat ditemukan pada bab 4.2.2.4. Oleh karena memenuhi batas treshold maka kita dapat memberi nilai pada predikat menjadi benar. Sehingga pada alur ketiga hasil triplet berbentuk subyek salah, predikat benar, dan obyek benar.

Untuk setiap pendeteksian alur 4, 5, 6 didapati benar semua tripletnya dikarenakan subyek ketiga alur tersebut terdapat pemanggilan metode milik kelas obyek nya. Lalu pada predikat benar dikarenakan terdapat metode itu didalam kelas obyeknya,

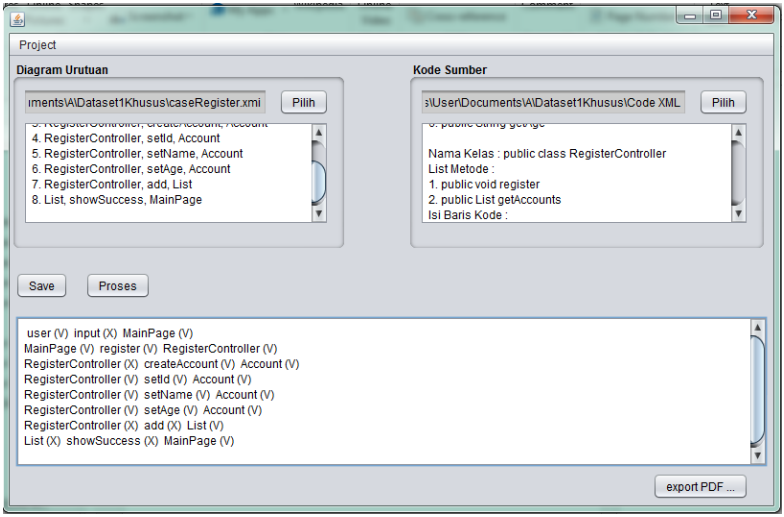
lalu obyek juga diberi nilai benar karena obyek merupakan sebuah kelas.

Setelah mendeteksi alur 4, 5, 6 dilanjutkan dengan mendeteksi alur ke 7 dimana alur ini subyek diberi nilai salah dikarenakan tidak terdapat pemanggilan add pada subyek. Lalu predikat diberi nilai salah juga dikarenakan tidak terdapat metode add pada obyek. Lalu untuk obyek diberi tanda benar dikarenakan obyek ditemukan kelasnya pada kode sumber.

Pada pendeteksian alur yang terakhir yaitu alur 8, didapati predikatnya merupakan sebuah reply atau create message sehingga langsung diberi nilai salah sehingga hasil dari triplet alur 8 ini adalah subyek salah dikarenakan tidak terdapat pemanggilan metode nya pada obyek, predikat diberi nilai salah dikarenakan predikat sebuah reply atau create message dan obyek diberi nilai salah dikarenakan obyek tidak ditemukan kelas pada kode sumber.

Dari penjelasan diatas hasil dari penjelasan tersebut adalah

1. subyek  benar, predikat salah,obyek benar
2. subyek benar, predikat benar, obyek benar
3. subyek salah, predikat benar, obyek benar
4. subyek benar, predikat benar, obyek benar
5. subyek benar, predikat benar, obyek benar
6. subyek benar, predikat benar, obyek benar
7. subyek salah, predikat salah, obyek benar
8. subyek salah, predikat salah, obyek salah




Dari hasil diatas berikut apabila dibandingkan dengan hasil dari kakas bantu dapat kita simpulkan bahwa kakas bantu menghasilkan hasil yang benar terhadap penilaian analisis secara manual.

## D.2 Skenario kode sumber yang berbeda

Dari sub-bab kedua lampiran ini, akan ditunjukkan hasil dari analisis kakas bantu apabila mendeteksi diagram urutan terhadap kode sumber yang berbeda.Didapati diagram urutan berupa kasus management library yang telah digambarkan pada sub-bab pertama lampiran ini.

Didapati kode sumber yang dimasukkan ke kakas bantu adalah kode sumber kasus transaction. Didapati kode sumber digambarkan secara berturut-turut:

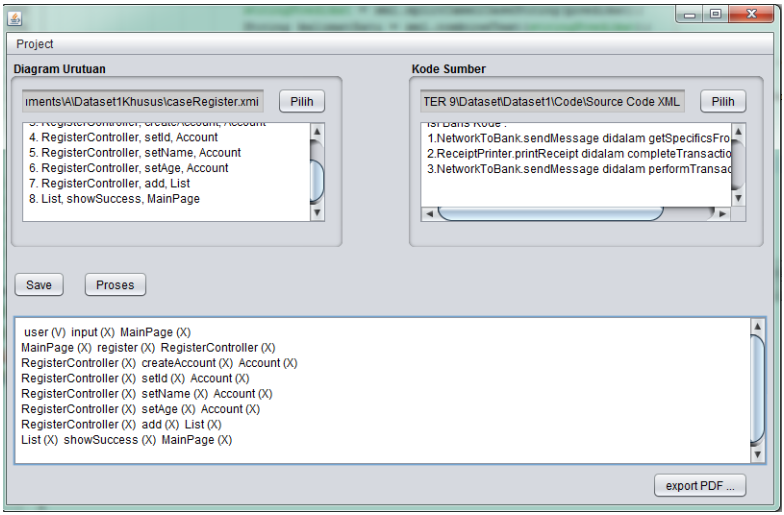
```
01. public abstract class Transaction
02. {
03.     private void getSpecificsFromCustomer(){
04.         ntb.sendMessage(message,balance);
05.     }
06.     private void completeTransaction(){
07.         rp.printReceipt();
08.     }
09.     public int getSerialNumber()
10.     {
11.     }
12. }
```

```
01. public class Log
02. {
03.     public void logSend(Message message)
04.     {
05.         Simulation.printLogLine("Message:   " + message.toString());
06.     }
07.     public void logResponse(Status response)
08.     {
09.         Simulation.printLogLine("Response:  " + response.toString());
10.     }
11.     public void logCashDispensed(Money amount)
12.     {
13.         Simulation.printLogLine("Dispensed: " + amount.toString());
14.     }
15.     public void logEnvelopeAccepted()
16.     {
17.         Simulation.printLogLine("Envelope:  received");
18.     }
19. }
```

```
01. public class ReceiptPrinter
02. {
03.     public void printReceipt(Receipt receipt)
04.     {
05.         Enumeration receiptLines = receipt.getLines();
06.         while (receiptLines.hasMoreElements())
07.         {
08.             Simulation.getInstance().printReceiptLine(
09.                 ((String) receiptLines.nextElement()));
10.         }
11.     }
12. }
```

```
01. public abstract class Transaction
02. {
03.     private void getSpecificsFromCustomer(){
04.         ntb.sendMessage(message,balance);
05.     }
06.     private void completeTransaction(){
07.         rp.printReceipt();
08.     }
09.     public int getSerialNumber()
10.     {
11.     }
12. }
```

Dari hasil kakas bantu ini mendeteksi setiap kasus ini, kakas bantu menemukan tidak ada satupun triplet yang mendapatkan nilai benar. Hasil kakas bantu dapat dilihat pada bagian dibawah ini:




Dari kesimpulan yang didapatkan, Kakas bantu memenuhi kebutuhan utama nya yaitu mendeteksi ketidaksesuaian.