



TESIS - KI142502

PROSES REFACTORING PAKET MENGGUNAKAN TEKNIK CLUSTERING

RATIH NINDYASARI

NRP 5111 201 028

DOSEN PEMBIMBING

Dr.Ir. Siti Rochimah, MT

PROGRAM STUDI MAGISTER

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2017

[Halaman ini sengaja dikosongkan]



THESIS - KI142502

**PACKAGE REFACTORING PROCESS USING
CLUSTERING TECHNIQUE**

RATIH NINDYASARI

NRP 5111 201 028

SUPERVISOR

Dr.Ir. Siti Rochimah, MT

MASTER PROGRAM

DEPARTEMENT OF INFORMATICS

FACULTY OF INFORMATION TECHNOLOGY

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2017

[Halaman ini Sengaja Dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:

RATIH NINDYASARI
Nrp. 5111 201 028

Dengan judul:

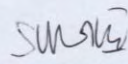
Proses Refactoring Paket Menggunakan Teknik Clustering

Tanggal Ujian : 9-1-2017

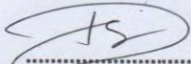
Periode Wisuda : Maret 2017

Disetujui oleh :

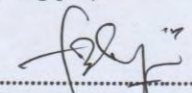
Dr. Ir. Siti Rochimah, M.T
NIP. 19681002 199403 2001


.....
(Pembimbing)

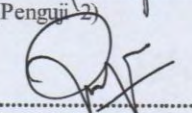
Daniel Oranova Siahaan, S.Kom., M.Sc. PD.Eng
NIP. 19741123 200604 1001


.....
(Penguji 1)

Fajar Baskoro, S.Kom., M.T.
NIP. 19740403 199903 1002


.....
(Penguji 2)

Risky Januar Akbar, S.Kom, M.Eng
NIP. 19870103 201404 1001


.....
(Penguji 3)

an. Direktur Program Pascasarjana
Asisten Direktur

Direktur Program Pascasarjana,

Prof. Dr. Ir. Tri Widjaja, M.Eng
NIP. 19611021 198003 1 000

Prof. Dr. Ir. Djauhar Manfaat, M.Sc. Ph.D.
NIP. 1960012021987011001



[Halaman ini sengaja dikosongkan]

PROSES REFACTORING PAKET MENGGUNAKAN TEKNIK CLUSTERING

Nama Mahasiswa : Ratih Nindyasari
NRP : 5111201028
Pembimbing : Dr.Ir. Siti Rochimah, MT.

ABSTRAK

Salah satu tantangan terbesar dalam persoalan Rekayasa Perangkat Lunak adalah ketika dihadapkan pada kondisi adanya kompleksitas internal perangkat lunak. Salah satu cara yang dapat digunakan untuk mengatasi persoalan kompleksitas pada perangkat lunak adalah dengan melakukan *refactoring*. *Refactoring* perangkat lunak merupakan sebuah teknik untuk melakukan perubahan struktur internal perangkat lunak tanpa merubah perilaku eksternal dari perangkat lunak itu sendiri. Banyak bagian dari perangkat lunak yang dapat direfactoring, salah satunya adalah paket. Proses *refactoring* pada level paket dilakukan dengan tujuan untuk meningkatkan hubungan keterkaitan (interdependensi) kelas-kelas dalam satu paket (*intra package cohesion*). kelas-kelas yang sebelumnya sudah berada dalam paketnya masing-masing akan di restrukturisasi. Kelas-kelas akan dikelompokkan menjadi satu paket berdasarkan pada hubungan kedekatan. Hubungan kedekatan antar kelas ini yang digunakan sebagai ukuran (*similarity measure*) antara kelas satu dengan kelas lainnya, sehingga kelas yang memiliki hubungan kedekatan tinggi akan ditempatkan dalam satu kelompok paket. Untuk mendukung proses pengelompokan kelas-kelas ini diperlukan suatu teknik yang dikenal dengan *clustering*. Metode yang digunakan untuk melakukan proses pengelompokan kelas-kelas dengan menggunakan metode SLINK (*Single Linkage*) dengan harapan akan memberikan hasil *cluster* baru yang akan berpengaruh pada peningkatan kohesi paket.

Kata kunci: *Clustering*, Paket, *Refactoring*, SLINK, Ukuran Similaritas

[Halaman ini sengaja dikosongkan]

PACKAGE REFACTORING PROCESS USING CLUSTERING TECHNIQUE

Student Name : Ratih Nindyasari
NRP : 5111 201 028
Advisor : Dr. Ir. Siti Rochimah, M.T.

ABSTRACT

One of the big challenges on Software Engineering is when we faced with the internal complexity. The way that can be used to overcome it is to do refactoring process. Refactoring is a technique to make changes internal structure without changing external behaviour. Many part of software can be refactoring, one of them is refactoring at the level package. The aims refactoring at the level package is to improve intra package relation, its call with package cohesion. classes that previously located in the package will be restructured. The classes will be grouped into one package based on closes relation. The close Relation between classes is used as a measure (similarity measure) between pair class, so classes that have high close relation will be placed in a one package. To support of the process, required a technique known as clustering. The method used to carry out the grouping process class by using SLINK (Single Linkage) with the hope of a new cluster can giving results that impact on improving cohesion package.

Keywords: Clustering, Package , Refactoring, SLINK, Similarity Measurement

[Halaman Ini Sengaja Dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT sehingga buku tesis ini dapat diselesaikan dengan baik. Meski dalam menyelesaikan buku ini banyak ditemui kesulitan, namun berkat bantuan dan bimbingan berbagai pihak, akhirnya Penulis berhasil menyelesaikan buku ini. Pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang membantu penulis dalam penulisan buku tesis ini sebagai berikut.

1. Kepada keluarga Penulis, Bapak, Ibu yang telah senantiasa mendoakan memberikan pengertian, dukungan, dan pengorbanan yang besar selama penulisan buku ini.
2. Kepada keluarga Riska Arinta yang telah senantiasa memberikan bantuan, tempat yang layak dan pengorbanan selama proses penyelesaian tesis.
3. Kepada Dosen Wali sekaligus Dosen Pembimbing ibu Dr. Ir. Siti Rochimah, M.T yang telah memberikan ilmu dan dengan sangat sabarnya membimbing selama menyelesaikan tesis ini.
4. Kepada para Dosen Penguji, Bapak Daniel Oranova, Bapak Fajar Baskoro dan Bapak Rizky Januar yang telah memberikan masukan berharga untuk menjadikan tesis ini lebih baik dari sebelumnya.
5. Kepada Dekan FTIF, Bapak Prof. Dr. Agus Zainal Arifin, yang telah memberikan motivasi, saran dan masukan agar Penulis membulatkan tekad untuk menyelesaikan tesis ini.
6. Kepada Kaprodi Bapak Waskito Wibisono, Ph.D yang telah memberikan kemudahan jalan untuk menyelesaikan tesis ini.
7. Kepada teman seperjuangan dan seangkatan 2011 Riska, Ratna, Pak Hengki, Pak andi, billy, dan mbak sari yang telah menjadi teman diskusi yang baik sekaligus sebagai tempat untuk mencari motivasi dan membantu menyelesaikan tesis ini.

8. Kepada staf administrasi Program Pascasarjana Teknik Informatika, Bu Lina dan Bu Rini, atas pengertian dan kebijaksanaannya dalam proses pengurusan administrasi.
9. Kepada pihak-pihak yang mendukung dan membantu penulis selama menyelesaikan tesis seperti: para driver gojek, widi, delivery gofood dan driver gozar.

Akhirnya, Penulis berharap, buku laporan tesis ini dapat memberikan kontribusi ilmiah bagi khasanah pengembangan riset di bidang Rekayasa Perangkat Lunak.

Surabaya, 21 Januari 2017

Ratih Nindayasari

DAFTAR ISI

LEMBAR PENGESAHAN	v
ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xv
DAFTAR TABEL	xvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	3
1.3 Tujuan Penelitian	3
1.4 Manfaat Penelitian	4
1.5 Batasan Penelitian	4
BAB II KAJIAN PUSTAKA	6
2.1 Refactoring Perangkat Lunak	6
2.2 Clustering	7
2.3 SLINK (Single Linkage)	7
2.4. Analisis Cluster	8
2.3 Entitas dan Atribut	9
2.3.1. Matriks Atribut Entitas	10
2.4. Ukuran Similaritas	11
2.5. Framework Ukuran Paket	12
2.5.1. Definisi Paket	12
2.5.2. Definisi Relasi	13

2.5.3. Ukuran Kohesi Paket.....	15
BAB III METODOLOGI PENELITIAN	19
3.1 Studi Literatur.....	19
3.2. Perancangan Kerangka Kerja	19
3.2.1 Pengumpulan Data dan <i>Preprocessing</i>	20
3.2.2 <i>Clustering</i>	22
3.2.3 Visualisasi dan Analisis	22
3.3 Pengujian dan Evaluasi	23
BAB IV HASIL PENELITIAN DAN PEMBAHASAN.....	25
4.1 Implementasi.....	25
4.1.1 Persiapan Data.....	25
4.1.2 Hubungan Antar Elemen Dalam Paket	26
4.2. Pengujian	31
4.2.1 Pengujian Kohesi Paket	32
4.2.2 Pengujian variansi <i>cluster</i>	36
4.3. Analisis dan Pembahasan	36
4.3.1. Analisis Hasil Pengujian nilai Kohesi.....	36
4.3.2. Analisis Hasil Pengujian Variansi <i>Cluster</i>	39
BAB V KESIMPULAN DAN SARAN.....	41
5.1 Kesimpulan	41
5.2 Saran	41
DAFTAR PUSTAKA	43

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi Contoh Program Dalam Diagram UML	Error! Bookmark not defined.
Gambar 3.1 Langkah-langkah Penelitian	19
Gambar 3.2 Proses Refactoring.....	20
Gambar 4.1 Dependensi Elemen Paket Dalam Diagram UML	26
Gambar 4.2 Diagram Batang Perbandingan Nilai Kohesi	37

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 2.1 Matriks Atribut Entitas	11
Tabel 2.2. Tipe Pasangan Entitas dan Atribut Besera Indikasinya	11
Tabel 4.1 Daftar Elemen Paket Sistem Trama	27
Tabel 4.2 Relasi Antar Kelas Dalam Sistem Trama	27
Tabel 4.3 Coeff Matriks Similaritas Antar Kelas Dalam Sistem Trama.....	30
Tabel 4.4 Hasil Pengelompokan Dengan Metode Single Linkage dengan 2 Cluster	30
Tabel 4.5 Hasil Pengelompokan Dengan Metode Single Linkage dengan 3 Cluster	30
Tabel 4.6 Hasil Pengelompokan Dengan Metode Single Linkage dengan Cutoff 0.8	31
Tabel 4.7 Nilai Kohesi Awal (Sebelum Pengelompokan)	33
Tabel 4.8 Nilai Kohesi Setelah Pengelompokan dengan Single Linkage dengan hasil 2 Cluster.....	34
Tabel 4.9 Nilai Kohesi Setelah Pengelompokan dengan Single Linkage Dengan Hasil 3 Cluster mapun Cutoff 0.8	35
Tabel 4.10 Tabulasi Nilai Kohesi dari Ketiga Pengujian	36

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Bab pendahuluan ini akan mengulas tentang apa yang menjadi latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian dan batasan penelitian.

1.1 Latar Belakang

Salah satu proses dalam daur hidup sebuah perangkat lunak adalah melakukan proses perawatan. Seorang analis, *programmer* ataupun profesional yang melakukan pembangunan perangkat lunak sering kali menghabiskan separuh dari waktunya untuk mempelajari, menganalisis, serta memahami lebih detail perangkat lunak yang dibangunnya. *Class* (kelas) merupakan salah satu elemen dari *source code* (kode program) yang penting untuk diperhatikan dan dimengerti, karena merupakan suatu bentuk tindakan dari proses perawatan perangkat lunak. Salah satu bentuk aktivitas merawat perangkat lunak dapat dilakukan dengan melakukan pengelompokan kelas kedalam modul atau sering disebut dengan istilah *package* (paket). Di dalam sebuah paket terdiri atas kelas-kelas yang saling berelasi baik dengan kelas-kelas lain dalam satu paket (*cohesi*) maupun berelasi dengan kelas-kelas lain yang ada diluar paket (*coupling*).

Selanjutnya ketika membahas tentang persoalan yang ada dalam Rekayasa Perangkat Lunak adalah ketika dihadapkan pada keadaan atau kondisi adanya kompleksitas internal perangkat lunak. Salah satu cara yang dapat digunakan untuk mengatasi persoalan kompleksitas pada perangkat lunak adalah dengan melakukan *refactoring*. *Refactoring* merupakan sebuah teknik untuk melakukan perubahan struktur internal perangkat lunak tanpa harus merubah perilaku eksternal dari perangkat lunak itu sendiri [1]. Teknik *refactoring* juga dapat dikatakan sebagai bentuk evolusi perangkat lunak. Beberapa tujuan dari teknik *refactoring* itu sendiri yaitu untuk memperbaiki desain perangkat lunak, membuat perangkat lunak agar lebih mudah dimengerti dari segi struktur ataupun adanya duplikasi-duplikasi pada kode, membantu menemukan *bugs* dikarenakan dengan *refactoring* dapat meningkatkan pemahaman terhadap kode dan tentunya akan sangat membantu dalam menemukan *bugs*. Berdasarkan pada beberapa tujuan

teknik *refactoring* ini, khusus dalam penelitian kali ini dititik beratkan untuk memperbaiki desain perangkat lunak dengan memperhatikan sisi *cohesion*.

Selama ini, beberapa aktivitas *refactoring* perangkat lunak dapat dilakukan pada kode sumber, *tools* (kakas bantu) maupun selain kode sumber seperti, kelas. Selain itu *refactoring* juga dapat dilakukan pada level *package* (paket) seperti yang akan dibahas secara detail pada penelitian kali ini. Sebuah paket biasanya dapat terdiri atas elemen-elemen seperti kelas, *interface* maupun sub paket, dimana elemen-elemen seperti kelas atau *interface* saling terkait satu dengan yang lainnya. Penyusunan kelas-kelas kedalam suatu modul atau paket juga penting untuk diperhatikan, karena akan mempengaruhi kualitas paket. Kualitas suatu paket dilihat berdasarkan pada nilai kohesi dan kopling. Kohesi paket harus semaksimal mungkin dan nilai kopling harus minimal. Pada penelitian kali ini difokuskan untuk meningkatkan nilai kohesi paket setelah dilakukan proses *refactoring*. Karena penempatan atau pengelompokan kelas-kelas ke dalam suatu paket ini penting, untuk itu diperlukan suatu Proses *refactoring* pada level paket, tujuannya adalah untuk meningkatkan hubungan keterkaitan (interdependensi) kelas-kelas dalam satu paket (*intra package cohesion*). Suatu sistem perangkat lunak memiliki sekumpulan paket-paket. Tiap paket terdiri dari beberapa kelas. Kembali fokus pada tujuan untuk meningkatkan kohesi, kelas-kelas yang sebelumnya sudah berada dalam paketnya masing-masing akan di restrukturisasi. Proses restrukturisasi kelas-kelas kedalam suatu paket inilah yang dikatakan sebagai proses *refactoring* pada level paket. Kelas-kelas akan dikelompokkan menjadi satu paket berdasarkan pada hubungan kedekatan. Hubungan kedekatan antar kelas ini yang digunakan sebagai ukuran (*similarity measure*) antara kelas satu dengan kelas lainnya, sehingga kelas yang memiliki hubungan kedekatan tinggi akan ditempatkan dalam satu kelompok paket. Untuk mendukung proses pengelompokan kelas-kelas ini maka diperlukan suatu teknik yang dikenal dengan *clustering*.

Dalam beberapa penelitian sebelumnya teknik *clustering* ternyata sudah sering digunakan untuk mendukung proses *refactoring*, diantaranya adalah sebagai berikut: Penelitian [2] menggunakan teknik *clustering Agglomerative* yaitu SLINK, CLINK dan WPGMA untuk mengidentifikasi kondisi kohesi pada

sebuah kelas yang rendah. Gupta dan Kaur [3] melakukan dekomposisi perangkat lunak dengan metode *agglomerative*, K-Means, KNN dan A-KNN. Pada tahun yang sama Alkhalid juga membandingkan metode *agglomerative* dengan A-KNN untuk melakukan *refactoring* pada tingkatan yang berbeda yaitu kelas dan paket [4][5]. Srinivas pada tahun 2013 [6] mencoba untuk menggunakan fungsi similaritas *hybrid XNOR* untuk *refactoring* struktur komponen. Hasilnya, dengan mengimplementasikan fungsi similaritas ini menghasilkan sekumpulan kelompok komponen dengan kohesi yang tinggi.

Berbekal dari studi literatur berdasarkan pada penelitian sebelumnya dan dari beberapa referensi lainnya, maka pada penelitian kali ini diusulkan untuk menerapkan teknik *clustering* untuk membantu melakukan proses *refactoring* pada paket. Diharapkan dengan menggunakan teknik *clustering* ini mampu membuktikan hipotesis penelitian kali ini yaitu hasil pengelompokan kelas-kelas kedalam paket baru atau kelompok baru dengan metode *single linkage* akan dapat meningkatkan kohesi paket. Hasil pengukuran kohesi paket lebih baik jika dibandingkan dengan kondisi dimana sebelum dilakukan proses restrukturisasi.

1.2 Perumusan Masalah

Perumusan masalah dalam penelitian kali ini dapat dijelaskan sebagai berikut ini:

1. Bagaimana menentukan entitas dan atribut yang akan digunakan sebagai elemen utama dalam proses pengelompokan kelas-kelas dalam modul/paket.
2. Bagaimana cara melakukan pengelompokan kelas dengan menggunakan metode SLINK dan membuktikan mampu meningkatkan kohesi paket.
3. Bagaimana cara melakukan pengukuran kohesi paket.

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Membuktikan bahwa teknik *clustering* mampu mengatasi persoalan pengelompokan pada level paket.

2. Melakukan pengukuran nilai kohesi paket sebelum dan sesudah proses *refactoring*.

1.4 Manfaat Penelitian

Dengan melaksanakan penelitian ini diharapkan dapat memberikan manfaat bagi perkembangan ilmu pengetahuan dalam lingkup pemeliharaan perangkat lunak, berkaitan dengan persoalan analisis *refactoring* perangkat lunak. Menindaklanjuti penelitian sebelumnya dengan menambahkan kontribusi dengan menerapkan teknik clustering pada level paket dan menunjukkan bagaimana cara melakukan pengukuran kohesi paket baik sebelum maupun setelah proses *refactoring*.

1.5 Batasan Penelitian

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut ini:

1. *Refactoring* dilakukan pada level paket, dengan kondisi bahwa paket yang ada dalam sebuah sistem perangkat lunak memiliki elemen berupa : sub paket, kelas ataupun *interface*.
2. Data Uji pada penelitian ini adalah data yang digunakan pada penelitian sebelumnya [5] yaitu: TRAMA SYSTEM, agar dapat dengan mudah dilakukan perbandingan.
3. Semua rangkaian proses *refactoring* pada penelitian ini belum dapat diimplementasikan secara otomatis dengan menghasilkan sebuah kakas bantu. Akan tetapi masih dilakukan dengan cara manual menggunakan beberapa kakas bantu seperti IntelliJ IDEA 2016.3 untuk mendapatkan data nilai nilai atribut yang akan diolah pada matriks similaritas nantinya dalam tahap pra proses *refactoring* dengan teknik *clustering*. Selanjutnya untuk melakukan teknik *clustering*, dilakukan dengan bantuan kakas bantu MATLAB.
4. Untuk membuktikan hasil pengelompokan elemen-elemen paket baik sebelum maupun sesudah proses *refactoring* dilakukan dengan melakukan pengujian untuk mengukur nilai kohesi tiap paket.

[Halaman ini Sengaja Dikosongkan]

BAB II

KAJIAN PUSTAKA

Pada bab ini akan dijelaskan mengenai teori-teori yang digunakan dalam melakukan penelitian. Teori yang terkait dapat diperoleh dari buku-buku ilmiah, laporan penelitian, karangan ilmiah, tesis dan disertasi, selain itu juga menyertakan sumber-sumber tertulis lainnya yang diperoleh melalui media internet. Beberapa teori pendukung penelitian pada laporan kali ini akan dijelaskan lebih rinci pada sub-sub bab berikut.

2.1 Refactoring Perangkat Lunak

Refactoring pertama kali dirumuskan oleh William F. Opdyke dalam disertasinya dan mulai dipraktekkan serta digunakan secara langsung setelah publikasi pada buku “Refactoring: Improve the Design of Existing Code”, yang ditulis oleh Martin Fowler pada tahun 1999. Sejak saat itu, *refactoring* merupakan teknik yang sering diterapkan untuk memperbaiki dan meningkatkan kualitas perangkat lunak.

Pendeteksian *refactoring* perlu dilakukan untuk mengetahui dimana *refactoring* harus dilakukan pada suatu perangkat lunak. Pada awal perkembangannya, pendeteksian ini dilakukan dengan identifikasi *bad smells* atau *code smells*. Dalam sebuah buku, Fowler dan Beck serta beberapa peneliti lainnya menetapkan daftar dari *bad smells* dan hubungannya pada beberapa teknik *refactoring*. Hal ini dilakukan untuk memudahkan pengambilan keputusan lokasi yang akan direfaktor. Berdasarkan hasil tersebut, secara umum saat ini ada kurang lebih 25 macam *bad smells* dan lebih dari 75 teknik *refactoring* pada *source code* perangkat lunak.

Refactoring perangkat lunak memiliki beberapa tahapan aktifitas sebagai berikut:

- a. Mengimplementasikan unit/ modul yang akan diuji pada program
- b. Mengidentifikasi bagian unit/ modul tersebut yang akan direfaktor
- c. Memilih teknik *refactoring* yang akan digunakan yang disesuaikan dengan kode *bad smells* yang teridentifikasi

- d. Mengimplementasikan teknik *refactoring* yang dipilih
- e. Mengimplementasikan pengujian secara regresi pada kode yang difaktor. Hal ini dilakukan untuk mengecek apakah fungsionalitas perangkat lunak yang diharapkan tidak berubah.
- f. Menilai dampak dari hasil *refactoring* yang dilakukan
- g. Melakukan perubahan yang saling menyesuaikan antara perangkat lunak (*source code*) dan artifak.

2.2 Clustering

Clustering adalah suatu metode pengelompokan berdasarkan ukuran kedekatan (kemiripan). *Clustering* beda dengan *group*, kalau *group* berarti kelompok yang sama, kondisinya kalau tidak sama sudah dapat dipastikan bukan kelompoknya. Tetapi kalau *cluster* tidak harus sama akan tetapi pengelompokannya berdasarkan pada kedekatan dari suatu karakteristik *sample* yang ada, salah satunya dengan menggunakan rumus jarak *eclidean*. *Clustering* juga bisa diartikan dengan suatu metode untuk mengelompokkan data dari entitas yang sama dalam satu kelompok dan entitas yang berbeda dalam kedalam kelompok lain. Entitas-entitas yang ada dalam satu *cluster* bersifat lebih erat dan dekat dibandingkan dengan entitas yang terdapat di dalam *cluster* yang berbeda [6]. Terdapat beberapa algoritma *clustering* salah satunya adalah *hierarchical agglomerative clustering* (*clustering* hierarkis). Di dalam algoritma *clustering* hierarki dapat menggunakan beberapa jenis algoritma seperti: *Single linkage* (SLINK), *complete linkage* (CLINK), *weighted pair group* (WPGMA).

2.3 SLINK (Single Linkage)

Metode *single linkage* merupakan salah satu dari beberapa teknik *clustering*. Metode ini termasuk dalam kategori hierarkikal *clustering*. SLINK dalam melakukan pengelompokan data menggunakan dasar jarak kedekatan. Sehingga setiap objek akan diukur jaraknya kemudian objek dengan jarak terdekatlah yang akan dijadikan satu *cluster*. Dalam metode SLINK setiap objek dianggap sebagai *single cluster*. Untuk mengimplementasikan metode SLINK ada beberapa langkah yang harus dilakukan [16] :

1. Mulai dengan N cluster, setiap cluster mengantong entitas tunggal dan sebuah matriks similaritas bertipe $N \times N$ yang berisi jarak.
2. Cari matriks jarak untuk pasangan cluster terdekat.
3. Gabungkan cluster yang terdekat. Kemudian beri label, kemudian update entitas pada matriks jarak dengan cara :
 - a. Hapus baris dan kolom yang bersesuaian
 - b. Tambahkan baris dan kolom yang memberikan jarak-jara antara cluster dengan cluster-cluster yang tersisa.
4. Ulangi langkah 2 dan 3 sebanyak $(N-1)$ kali.

2.4. Analisis Cluster

Analisis cluster digunakan untuk mengukur nilai hasil penyebaran data hasil clustering [16]. Untuk mengukurnya dilakukan dengan mengetahui nilai varian. Varian pada teknik clustering ada dua macam, yaitu :

- a. variance within cluster, yaitu tipe varian ini mengacu pada jarak antar anggota pada cluster yang sama.
- b. Variance between cluster, yaitu tipe varian yang mengacu pada jarak antar cluster.

Ada dua ketentuan apabila menentukan cluster yang ideal menggunakan cara perbandingan antara variance within cluster (V_w) dan variance between cluster (V_b). Berikut ini adalah persamaan yang dapat digunakan untuk menghitung V_w dan V_b :

- a. berdasarkan nilai minium

$$V = \frac{V_w}{V_B} \quad (1)$$

Dimana :

V = nilai variance

V_w = nilai variance within cluster

V_B = nilai variance between cluster

Cluster yang dianggap ideal adalah cluster yang memiliki nilai variance yang paling kecil.

b. Berdasarkan nilai maksimum

$$V = \frac{V_B}{V_w} \quad (2)$$

Cluster yang ideal adalah cluster yang memiliki nilai variance yang paling besar.

Sebelum mencari nilai variance (V), langkah yang harus dilakukan terlebih dahulu adalah mencari nilai V_w dan V_B . Berikut ini adalah bentuk persamaan untuk menentukan nilai V_w dan V_B :

$$V_w = \frac{1}{N-k} \sum_{i=1}^k (n_i - 1) \cdot V_i^2 \quad (3)$$

Dimana :

N : jumlah semua data

K = jumlah cluster

n_i = jumlah data pada cluster ke -i

V_i^2 = variance pada cluster ke -i

Sebelum menghitung variance within perlu menghitung nilai V_c :

$$V_c^2 = \frac{1}{n_c-1} \sum_{i=1}^c (d_i - \bar{d}_i)^2 \quad (4)$$

Dimana :

V_c^2 = variance pada cluster c

$c = 1..k$, dimana k = jumlah cluster

n_c = jumlah data pada cluster c

d_i = data ke-i pada suatu cluster

\bar{d}_i = rata-rata dari data pada suatu cluster

Kemudian untuk mencari nilai V_B menggunakan persamaan :

$$V_B = \frac{1}{k-1} \sum_{i=1}^k n_i (\bar{d}_i - \bar{d})^2 \quad (5)$$

Dimana :

\bar{d} = rata-rata dari \bar{d}_i

2.3 Entitas dan Atribut

Entitas adalah objek yang akan dikelompokkan. Pada proses *refactoring* pada level paket, kelas atau *interface* dipilih sebagai entitas, karena didalam paket itu sendiri terdiri dari kelas-kelas yang digunakan untuk mendukung jalannya

sebuah program aplikasi. Supaya entitas-entitas dapat dikelompokkan, maka sifat-sifat dari entitas harus didapatkan. Sifat dari sebuah entitas ini disebut dengan atribut. Atribut digunakan untuk menghitung bagaimana kedekatan dari dua buah entitas. Entitas dikatakan memiliki kesamaan atau kemiripan apabila saling berbagi atribut. Oleh karena itu method dipilih sebagai atribut. Jadi entitas dapat dikelompokkan ke dalam kelompok-kelompok berdasarkan pada nilai atribut. Sehingga dalam proses *refactoring* nantinya, entitas-entitas akan direstrukturisasi supaya mendapatkan suatu penyusunan kelompok yang terbaik untuk mengelompokkan kelas-kelas dalam suatu paket-paket agar dapat meningkatkan kohesi intra paket.

Entitas dapat mengakses *method* dalam *class*. Entitas juga bisa mengakses *method* pada kelas lain dengan menggunakan *instance of class*. Nilai atribut adalah jumlah berapa kali kelas mengakses *method*. Dengan demikian, nilai atribut dapat diukur dengan menggunakan formula seperti dibawah ini:

$$v(Att_i, ent_j) = A(Att_i, ent_j); i \in [1 \dots a], j \in [1 \dots b] \quad (6)$$

Dimana $v(Att_i, ent_j)$, adalah nilai atribut untuk entitas ent_j ; a adalah jumlah *method* dalam sistem; b adalah jumlah *class* dalam sistem; $A(Att_i, ent_j)$ adalah jumlah dari akses ke *method* diwakili oleh Att_i , atribut dalam *class* diwakili oleh ent_j entitas. Semakin banyak atribut dari dua entitas saling digunakan, dikatakan semakin mirip. Jika semakin banyak atribut dari kedua entitas ini saling dibagi atau diakses atau dipakai, dapat dikatakan entitas ini semakin dekat. Jika entitas-entitas ini semakin banyak berbagi atribut, kedua entitas ini diindikasikan memiliki kedekatan secara fungsionalitas dan entitas ini harus di tempatkan atau diletakkan dalam paket yang sama supaya memaksimalkan kohesi.

2.3.1. Matriks Atribut Entitas

Matriks atribut entitas adalah suatu matriks dimana barisnya merepresentasikan nama kelas dan bagian kolom merepresentasikan atribut yaitu *method*. Nilai yang berada pada perpotongan antara baris dan kolom merupakan banyaknya atau jumlah *method* yang diakses dalam kelas. Salah satu contoh

potongan matriks atribut dan entitas dapat dilihat pada Tabel 2.1. Jika dilihat berdasarkan pada Tabel 2.1 terlihat bahwa Method A diakses sebanyak dua kali didalam kelas 2. Karena titik perpotongan antara baris dan kolomnya merupakan hasil hubungan atau keterkaitan antara entitas yang berpasangan maka setiap dua entitas ini memiliki tiga tipe kombinasi pasangan nilai atribut yang berbeda-beda untuk setiap atributnya. Daftar kombinasi pasangan ini dapat dibaca pada Tabel 2.2.

Tabel 2.1 Matriks Atribut Entitas

Entitas	Atribut			
	MethodA	MethodB	MethodC	MethodD
Class1	0	0	0	0
Class2	2	0	0	0
Class3	0	0	0	0
Class4	0	0	0	0

Tabel 2.2 Tipe Pasangan Entitas dan Atribut Beserta Indikasinya

Tipe Kombinasi	Indikasi
n-0/0-n	Atribut berada pada satu entitas dan tidak ada pada entitas yang lain
n-m	Atribut sama-sama berada pada kedua pasangan entitas.
0-0	Atribut tidak berada pada kedua entitas.

Kombinasi tipe entitas yang berpasangan ini memiliki tiga jenis, untuk tipe yang pertama adalah tipe n-0/0-n, artinya atribut yang digunakan untuk menentukan hubungan keterkaitan antara dua entitas ini muncul hanya di salah satu entitas. Dan tidak muncul di entitas yang kedua. Tipe ke dua adalah n-m, artinya atribut muncul di kedua entitas, tipe ini akan memberikan kontribusi terhadap hasil perhitungan Coeff. Similaritas. Kemudian tipe yang terakhir adalah 0-0, artinya tidak ada atribut yang muncul di kedua entitas.

2.4. Ukuran Similaritas

Teknik *clustering* bergantung pada pengukuran similaritas. Pengukuran similaritas ini menunjukkan derajat pengukuran kemiripan antara dua kelas atau dua entitas. Dalam satu sistem terdapat satu matriks similaritas. Ukuran similaritas

ini direpresentasikan dengan suatu koefisien kemiripan. Koefisien kemiripan akan bergantung pada entitas dan atribut. Untuk memperoleh koefisien kemiripan antara dua entitas dapat dilakukan perhitungan dengan menggunakan persamaan (2).

$$Coeff = \frac{Similarity\ factor}{Similarity\ factor + Dissimilarity\ factor} \quad (7)$$

Dimana :

$$Similarity\ factor = \sum_{k=0}^a \min(n_k, m_k) \quad (8)$$

Dimana :

a: jumlah kecocokan antara dua entitas.

$$Dissimilarity\ factor = \text{jumlah kecocokan } n - 0 / 0 - n \text{ antara dua entitas} \quad (9)$$

2.5. Framework Ukuran Paket

2.5.1. Definisi Paket

Definisi dari paket adalah sekumpulan dari elemen-elemen yang dapat terdiri dari kelas, *interface* atau juga paket lainnya yang dapat disebut dengan istilah sub paket yang didalamnya juga saling berelasi satu sama lain. Paket yang memiliki sub paket didalamnya akan membentuk sebuah struktur hierarki paket. Sebuah paket yang berada pada level i dapat direpresentasikan sebagai $p^i = \langle E^{i+1}, R^{i+1} \rangle$ merupakan sekumpulan elemen paket dari paket p^i pada level $i+1$, dan R^{i+1} adalah sekumpulan relasi pada E^{i+1} pada struktur hierarki level $i+1$. Relasi pada sekumpulan elemen ini berbentuk biner yang merupakan hubungan langsung atau relasi langsung antara pasangan elemen yang terdapat dalam paket. Keterkaitan atau relasi elemen dalam paket ini yang menyebabkan ketergantungan intra paket dalam sebuah paket yang sebut dengan kohesi. selama pengukuran kohesi dari sebuah paket, yang dipertimbangkan hanya kekompakan yang terjadi antara unsur-unsurnya (kelas, *interface* atau subpaket) dan tidak memperhitungkan kohesi individual dari elemen. Berikut ini adalah penjelasan dari masing-masing bentuk paket.

- Subpaket

Dalam lingkup sebuah sistem, beberapa paket dapat disimbolkan dengan p^i . Kemudian jika didalam paket (p^i) ini terdapat elemen paket lagi maka dapat ditambahkan level $i+1$ dan terbentuk menjadi hierarki. Sebuah paket yang berada pada kondisi $p_1^{i+1} = E_1^{i+2}, R_1^{i+2}$ dapat dikatakan mempunyai subpaket berupa paket $p_2^i = E_2^{i+1}, R_2^{i+1}$ apabila $p_1^{i+1} \in E_2^{i+1}, p_1^{i+1} = \text{subP}(p_2^i)$. Level teratas dalam sebuah struktur hierarki paket adalah paket yang bukan merupakan subpaket, dan level terendahnya adalah apabila paket tersebut sudah tidak memiliki subpaket.

- *Disjoint* Paket

Disjoint paket adalah semua paket yang secara struktur hierarki berada pada level yang sama. Misalnya terdapat dua buah paket yaitu p_1^i dan p_2^i yang berada pada level i . *Disjoint* paket ini selain berada pada level paket yang sama, juga tidak pernah berbagi kelas-kelas maupun *interface*. Sehingga paket ini bisa dikatakan sebagai *single* paket yang merupakan bagian dari struktur hierarki paket.

- *Empty* Paket

Paket ini tidak memiliki elemen, dan tidak juga memiliki relasi. Sehingga disebut juga dengan paket kosong dan direpresentasikan dengan (\emptyset, \emptyset) .

2.5.2. Definisi Relasi

Relasi atau hubungan antara elemen-elemen yang berpasangan pada sebuah paket dapat berupa tipe relasi *inheritance*, *agregasi*, maupun *reference*. Tipe-tipe relasi ini merupakan bentuk hubungan berpasangan antara elemen-elemen e_1^i dan e_2^i dari sebuah paket yang memiliki bentuk struktur hierarki. Relasi direpresentasikan dengan $r(e_1^i, e_2^i)$. Relasi yang ada antara dua buah elemen dinotasikan dengan $r(e_1^i, e_2^i) = 1$ atau $e_1^i \rightarrow e_2^i$. Relasi antara elemen-elemen ini adalah asimetris sehingga dapat dikatakan relasi $e_1^i \rightarrow e_2^i$ tidak berarti bahwa sama dengan relasi $e_2^i \rightarrow e_1^i$. Dengan adanya tipe elemen yang bermacam-macam dalam sebuah paket seperti kelas, *interface*, dan subpaket akan menyebabkan tipe relasi yang berbeda juga diantara tiap elemen-elemen. Untuk

tipe elemen *interface* dan kelas dapat dikatakan memiliki perilaku yang sama sehingga dapat dikatakan apabila memiliki tipe relasi yang sama juga. Berikut ini adalah beberapa jenis relasi yang dapat terjalin dari elemen-elemen yang berada dalam suatu paket. Relasi yang terjalin dapat berasal dari relasi kelas-kelas, paket-paket, sub paket dengan kelas dan relasi yang terakhir terjadi antara kelas dengan sub paket, Masing-masing relasi yang terjadi anantara elemen-elemen dalam suatu paket akan dijelaskan secara detail adalah sebagai berikut:

- Relasi kelas-kelas

Jenis tipe relasi ini terbentuk antara pasangan elemen kelas dengan kelas atau dengan *interface* dari sebuah paket. Tipe relasi ini dapat berupa *inheritance*, *agregasi*, maupun *reference*. Relasi antara elemen yang berpasangan pada sebuah paket dapat dituliskan dalam bentuk sebagai berikut,

$$p^i (p^i = (E^{i+1}, R^{i+1}) \quad \text{dimana}$$

$r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1} \& e_1^{i+1}, e_2^{i+1} \in E^{i+1}$ direpresentasikan sebagai

$$r(e_1^{i+1}, e_2^{i+1}) = 1 \mid$$

$(r(c_1^{i+1}, c_2^{i+1}) = 1 \text{ atau } c_1^{i+1} = e_1^{i+1} \text{ atau } c_2^{i+1} = e_2^{i+1})$. Jenis relasi

kelas-kelas sama halnya dapat diterapkan pada jenis relasi *interface*-kelas, kelas-*interface* ataupun *interface-interface*.

- Relasi paket-paket

Tipe relasi ini terjadi antara dua buah subpaket pada sebuah paket. Sebuah paket dikatakan memiliki relasi dengan paket lainnya jika elemen-elemen dalam sebuah paket tersebut mempunyai satu satu buah atau lebih relasi dengan elemen-elemen di paket lainnya.

- Relasi sub paket-kelas

Tipe relasi ini adalah bentuk hubungan antara sub paket dengan kelas atau *interface* pada sebuah paket. Sebuah sub paket yang berada pada (level $i+1$) dari suatu paket level i dapat dikatakan juga menjadi relasi untuk sebuah kelas atau *interface* yang berada pada level $i+2$ dari sebuah subpaket.

- Relasi kelas-sub paket

Sebuah relasi kelas-sub paket terjadi dari bentuk hubungan dari sebuah kelas (*interface*) dengan subpaket pada sebuah paket. Kelas dikatakan

memiliki koneksi dengan subpaket, jika terdapat relasi dari kelas yang berada pada level $i+1$ ke beberapa kelas (*interface*) yang berada pada level $i+2$ atau setingkat di atasnya pada sebuah sub paket yang berada pada level $i+1$ pada sebuah paket p^i ($p^i = (E^{i+1}, R^{i+1})$) dimana $r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1}$ & $e_1^{i+1}, e_2^{i+1} \in E^{i+1}$.

2.5.3. Ukuran Kohesi Paket

Suatu bagian dari sebuah bentuk struktur hierarki dari suatu paket dapat dilihat sebagai sekumpulan elemen dan relasi antara pasangan elemen yang berada pada level hierarki berikutnya. Relasi antar elemen pada sebuah paket ini yang menyebabkan adanya dependensi intra paket. Dependensi ini antara elemen paket yang secara efektif dikatakan sebagai suatu kohesi dari sebuah paket. Selama mengukur kohesi pada sebuah paket, kohesi elemen-elemen ini hadir pada level hierarki selanjutnya dan tidak pada individual elemen-elemennya. Ukuran kohesi pada sebuah paket dapat dihitung dengan sebuah derajat kohesi antara elemen, dengan menggunakan sebuah rasio jumlah relasi dengan elemen yang berpasangan. Kohesi paket dapat didefinisikan sesuai dengan bentuk persamaan 5.

$$PCoh(p^i) = \begin{cases} 0 & \text{if } (n = 0) \\ \frac{\sum_{x=1}^n \sum_{y=1 \wedge y \neq x}^{n-1} r(e_x^{i+1}, e_y^{i+1})}{n(n-1)} & \text{if } (n > 1) \\ 1 & \text{if } (n = 1) \end{cases} \quad (10)$$

Dimana n adalah jumlah elemen (kelas, *interface*, subpaket) pada struktur hierarki level $i+1$ dalam sebuah paket p^i ($p^i = (E^{i+1}, R^{i+1})$) didefinisikan berada pada level i dan e_x^{i+1} dan e_y^{i+1} bersama membentuk elemen berpasangan pada sebuah paket p^i . Sedangkan relasi antara elemen dinotasikan dengan $r(e_x^{i+1}, e_y^{i+1})$ dimana $r(e_x^{i+1}, e_y^{i+1}) \in R^{i+1}$ dan $e_x^{i+1}, e_y^{i+1} \in E^{i+1}$. Sehingga $r(e_x^{i+1}, e_y^{i+1})$ merepresentasikan sebuah relasi antara pasangan elemen pada sebuah paket. Dua elemen paket dikatakan berelasi jika dan hanya jika paling sedikit memiliki satu relasi diantara elemennya dan relasi yang terjadi dapat bertipe *inheritance*,

agregasi, maupun *reference*. $\sum_{x=1}^n \sum_{y=1 \wedge y \neq x}^{n-1} r(e_x^{i+1}, e_y^{i+1})$ dapat dikatakan sebuah representasi dari jumlah bentuk binari, yang menandakan adanya relasi langsung (elemen yang berada pada struktur hierarki level $i+1$) antara semua elemen yang berada pada paket level i . Relasi $r(e_x^{i+1}, e_y^{i+1})$ bisa atau bisa tidak sama dengan relasi $r(e_y^{i+1}, e_x^{i+1})$.

Maksimum kemungkinan jumlah relasi diantara n elemen adalah $n*(n-1)$, yang merepresentasikan jumlah maksimum kemungkinan adanya dependensi intra paket untuk sebuah paket. Nilai kohesi untuk sebuah paket akan bernilai 0, jika tidak ada relasi diantara elemen-elemennya dan 1 jika setiap elemen pada sebuah paket itu berelasi dengan semua elemennya. 0 adalah nilai kohesi minimum dan 1 adalah nilai maksimum. Kasus khusus bisa terjadi jika $n=0$, artinya tidak ada elemen dan tidak terbentuk relasi juga dalam sebuah paket p^i . Sehingga paket p^i dapat dikatakan sebagai paket kosong $\langle \emptyset, \emptyset \rangle$ dan nilai kohesi untuk paket ini adalah 0, $PCoh(p^i) = 0$. Kemudian jika $n=1$, berarti bahwa paket p^i terdiri atas *single* elemen, dan kohesi paket p^i yang hanya memiliki sebuah elemen ini adalah 1. $PCoh(p^i) = 1$. Ilustrasi atau gambaran bentuk dari pengukuran kohesi paket dapat digambarkan seperti pada Contoh potongan program seperti yang terdapat pada referensi [15] dibawah ini adalah sebagai berikut:

Contoh potongan program untuk menghitung ukuran kohesi paket :

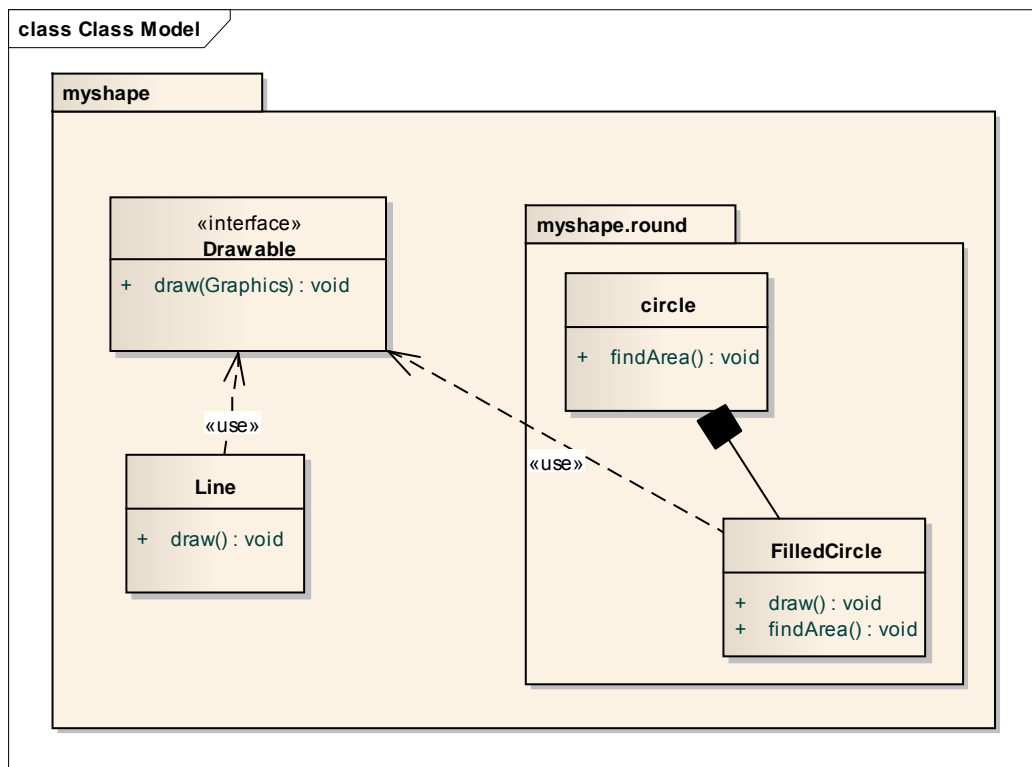
<pre>package myshapes; public interface Drawable { public void draw(Graphics g); } class Line implements Drawable { public void draw(Graphics g) { . . . // do something – presumably, draw a line } . . . // other methods and</pre>	<pre>package myshapes.round; import myshapes.Drawable; public class Circle { public void findArea() { . . . // find area of circle } . . . // other methods and variables } Class FilledCircle extends Circle implements Drawable{ public void draw(Graphics g) {</pre>
---	---

```

variables
}
... // do something – draw a filled Circle
}
void findArea() {
... // find area of filled circle
}
... // other methods and variables
}

```

Berdasarkan pada contoh potongan program diatas dapat direpresentasikan menjadi bentuk diagram UML seperti pada tertera pada Gambar 2.1. Relasi yang terdapat dalam diagram menunjukkan adanya hubungan langsung dari elemen satu dengan yang lainnya.



Gambar 2.1 Ilustrasi Contoh Program dalam Diagram UML

Berdasarkan pada contoh potongan program dapat dijelaskan bahwa paket "myshapes" memiliki tiga elemen dan dua relasi. Elemen-elemen ini adalah *interface* "Drawable", kelas "line", dan sub paket "round". Kemudian relasi yang terjalin diantara elemen-elemennya adalah kelas "line" dengan *interface*

“Drawable”, hal ini dikarenakan kelas “line” mengimplementasikan *interface* “drawable” dalam kelasnya. Dan kemudian untuk relasi berikutnya adalah antara sub paket “round” dan *interface* “Drawable”, hal ini dikarenakan keberadaan relasi antara kelas dengan elemen “FilledCircle” pada sub paket “round” dan *interface* “Drawable”. Jika paket “myshape” diasumsikan berada pada level 0, kemudian elemen-elemen yang berada di paket “myshape” berada hierarki level 1, maka dikatakan bahwa paket “round” berada pada level 1 dan elemen-elemen serta relasi didalamnya dianggap berada di hierarki level 2. Elemen dan relasi yang berada di level 2 dienkapsulasi didalam paket “round”. Sehingga dalam pengukuran kohesi paket “myshape” relasi yang terlihat adalah antara “filledCircle” (elemen pada sub paket “round”) dengan *interface* “Drawable” (elemen pada paket “myshape”). Atau dapat juga disebut sebagai relasi antara sub paket “round” dan *interface* “Drawable”.

Berdasarkan pada definisi dan penjelasan pada paragraf sebelumnya tentang elemen dan relasi yang terdapat dalam paket “myshape” maka dapat diketahui bahwa, $n=3$ dan $\sum_{x=1}^n \sum_{y=1 \wedge y \neq x}^{n-1} r(e_x^{i+1}, e_y^{i+1}) = 2$, sehingga kohesi paket “myshape” ketika berada pada hierarki level 0 adalah :

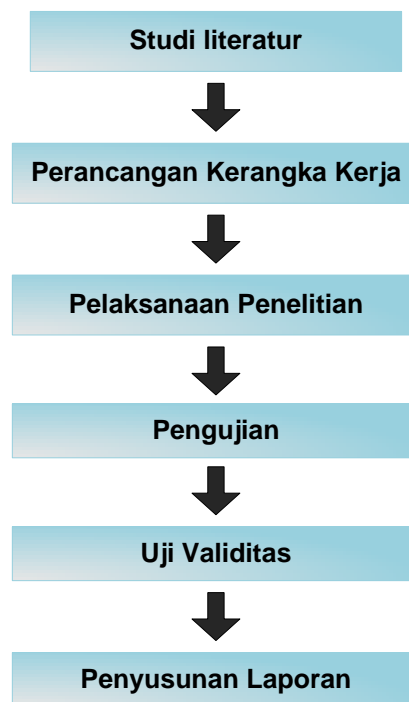
$$PCoh(myshape^0) = \frac{2}{3*(3-1)} = 0.334 .$$

BAB III METODOLOGI PENELITIAN

Secara umum, penelitian ini diawali dengan tahap studi literatur, perancangan kerangka kerja, implementasi, serta diakhiri dengan tahap uji coba. Secara detail, penelitian ini dirancang dengan urutan sebagai berikut:

3.1 Studi Literatur

Studi literatur merupakan fase untuk mempelajari referensi, teori, fakta, konsep *refactoring* perangkat lunak, paket, metode *clustering*, kohesi intra paket. Dari hasil studi literatur beberapa konsep tersebut ditemukan kekurangan dan kelebihan dari penelitian yang sudah pernah dilakukan sebelumnya.

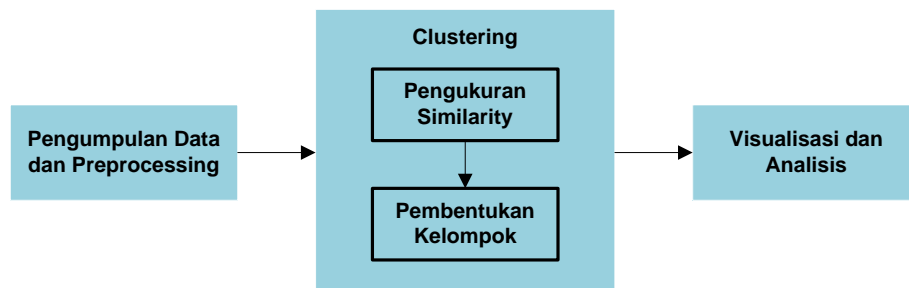


Gambar 3.1. Langkah-langkah Penelitian

3.2. Perancangan Kerangka Kerja

Pada bagian perancangan kerangka kerja ini akan menjelaskan tentang langkah-langkah yang harus dilakukan dalam proses *refactoring*. Pada penelitian

kali ini Proses *refactoring* yang akan dilakukan mengacu pada pendekatan proses *refactoring* yang telah dilakukan oleh Lung [9] dan dapat dideskripsikan pada Gambar 3.2. Pada gambar tersebut terdapat beberapa proses yang akan dilalui. Proses pertama adalah pengumpulan data dan *preprocessing*, kemudian dilanjutkan dengan proses kedua yaitu *clustering*, proses visualisasi dan analisis. Penjelasan tentang masing-masing proses akan lebih dijelaskan secara mendalam pada sub bagian berikutnya.



Gambar 3.2 Proses *Refactoring*

3.2.1 Pengumpulan Data dan *Preprocessing*

Pada proses pengumpulan data dan *preprocessing* ini akan dilakukan dua proses utama, yaitu :

- Pengumpulan data

Proses pengumpulan data adalah proses persiapan untuk menentukan *dataset* yang akan digunakan sebagai objek penelitian. *Dataset* yang digunakan berupa data paket, dan kelas yang diperoleh dari suatu *source code* program aplikasi. Program aplikasi yang digunakan adalah Trama [12]. Trama adalah sebuah program untuk menampilkan matriks dalam bentuk GUI yang dibangun dengan menggunakan bahasa pemrograman java.

- *Preprocessing*

Pada proses *preprocessing* ini dilakukan tahap *parsing* terhadap *source code* dari suatu program aplikasi. Proses ini dilakukan secara otomatis dengan menggunakan sebuah kakas bantu yaitu IntelliJ IDEA. Dengan

menggunakan kakas bantu ini akan membantu mengetahui struktur paket, elemen-elemen yang terdapat dalam sebuah paket seperti kelas, *interface* dan sub paket. Selain itu Intelij IDEA juga dapat digunakan untuk mengetahui interdependensi dari tiap-tiap elemen yang terdapat dalam sebuah paket. Berdasarkan pada hasil pengamatan interdependensi tiap elemen dari sebuah paket, dapat digunakan untuk melakukan proses selanjutnya yaitu menyusun data matriks. Dalam konteks penelitian kali ini data matriks digunakan untuk menentukan tingkat kedekatan antara elemen-elemen yang terdapat dalam sebuah paket. Misalnya terdapat elemen kelas, *interface* dan sub paket. Tentunya elemen kelas dengan kelas yang lain ada yang saling berhubungan atau berelasi. Elemen yang saling berhubungan ini membutuhkan sebuah patokan berupa suatu rentang nilai untuk mengetahui seberapa pastinya keterhubungan antar elemen satu dengan yang lainnya. Data matriks dapat juga berisi kelas dan method yang tersusun dalam bentuk matriks entitas dan atribut, sehingga disebut dengan matriks entitas-atribut. Data Matriks entitas dan atribut akan dijelaskan pada sub bab berikutnya.

- Data Matriks Entitas-Atribut

Data matriks ini adalah hasil keluaran dari proses *parsing*. Berbentuk matriks dua dimensi dengan entitas sebagai kolom dalam hal ini adalah kelas-kelas dan atribut sebagai baris yang dalam hal ini adalah method. Perpotongan antara entitas dan atribut berisi dinamakan nilai atribut, yaitu berisi berapa kali atau jumlah method yang telah diakses oleh suatu kelas.

- Entitas dan Atribut

Entitas adalah objek yang akan dikelompokkan. Pada proses *refactoring* pada level paket, kelas dipilih sebagai entitas, karena didalam paket itu sendiri terdiri dari kelas-kelas yang digunakan untuk mendukung jalannya sebuah program aplikasi. Supaya entitas-entitas dapat dikelompokkan, maka sifat-sifat dari entitas harus didapatkan. Sifat dari sebuah entitas ini disebut dengan atribut. Atribut digunakan untuk menghitung bagaimana kedekatan dari dua buah entitas. Entitas dikatakan memiliki kesamaan atau kemiripan apabila saling berbagi atribut. Oleh karena itu *method* dipilih

sebagai atribut. Jadi entitas dapat dikelompokkan ke dalam kelompok-kelompok berdasarkan pada nilai atribut. Sehingga dalam proses *refactoring* nantinya, entitas-entitas akan direstrukturisasi supaya mendapatkan suatu penyusunan kelompok yang terbaik. Tujuannya untuk mengelompokkan kelas-kelas dalam suatu paket-paket agar dapat meningkatkan kohesi intra paket. Selain *method*, terdapat atribut lainnya juga yang dapat digunakan untuk menentukan hubungan interdependensi antara elemen-elemen yang terdapat dalam sebuah paket. Atribut ini diantaranya adalah, variabel, tipe parameter, dsb.

3.2.2 Clustering

Proses lanjutan setelah menentukan entitas dan atribut yang digunakan sebagai komponen penyusun data matriks, tahap selanjutnya adalah melakukan proses *clustering*. Didalam Tahap *clustering* terdapat langkah yang paling penting dan fundamental, yang nantinya akan mempengaruhi proses *clustering* itu sendiri, yaitu pengukuran similaritas.

- **Pengukuran Similaritas**
Proses pengukuran similaritas dilakukan sebelum melakukan proses *clustering*. Pengukuran ini digunakan untuk mengukur kedekatan atau kemiripan antara dua entitas. Pada dasarnya, similaritas antara dua entitas ini dihitung berdasarkan pada jumlah atribut yang telah digunakan bersama-sama atau saling berbagi atribut oleh kedua entitas.
- **Pembentukan Kelompok**
Setelah mendapatkan nilai similaritas, dilanjutkan dengan pembentukan kelompok atau *cluster*. Karena nilai similaritas ini digunakan sebagai dasar pengelompokan. Pengelompokan dilakukan dengan menggunakan algoritma *agglomerative clustering* yaitu *Single Linkage*.

3.2.3 Visualisasi dan Analisis

Setelah proses *clustering* dilakukan langkah selanjutnya adalah menampilkan hasil *clustering* dalam bentuk dendrogram. Kemudian berdasarkan bentuk dendrogram tersebut dapat digunakan untuk melakukan proses analisis

terhadap hasil *clustering* yang baru. Berdasarkan pada hasil pengelompokan baru dapat digunakan untuk menyusun elemen-elemen seperti kelas, *interface* dan sub paket kedalam sebuah paket tertentu. *Clustering* digunakan untuk membantu mengelompokkan elemen-elemen tersebut. Elemen dikelompokkan berdasarkan nilai kedekatan antar elemen. Proses analisis dilakukan dengan cara membandingkan nilai keterkaitan antar elemen sebelum dilakukan proses *clustering* dan sesudah dikelompokkan kembali.

3.3 Pengujian dan Evaluasi

Pengujian dan evaluasi dilakukan untuk mengetahui ketepatan hasil dari metode yang telah digunakan. Adapun langkah-langkah pengujian yang akan dilakukan adalah dengan membandingkan jumlah keterkaitan kelas dalam satu paket dan antar paket sebelum proses *refactoring* dengan setelah terbentuk kelompok paket baru atau setelah dilakukan proses restrukturisasi kelas-kelas ke dalam paket baru. Dengan adanya proses restrukturisasi kelas-kelas kedalam kelompok paket yang baru diharapkan akan menghasilkan kohesi yang meningkat apabila dibandingkan sebelum dilakukan restrukturisasi paket. Pengukuran kohesi paket dilakukan dengan menghitung kohesi menggunakan persamaan 10.

[Halaman ini sengaja dikosongkan]

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini akan dijelaskan tentang tahapan implementasi, pengujian dan analisis hasil penelitian berdasarkan metodologi yang telah dibangun.

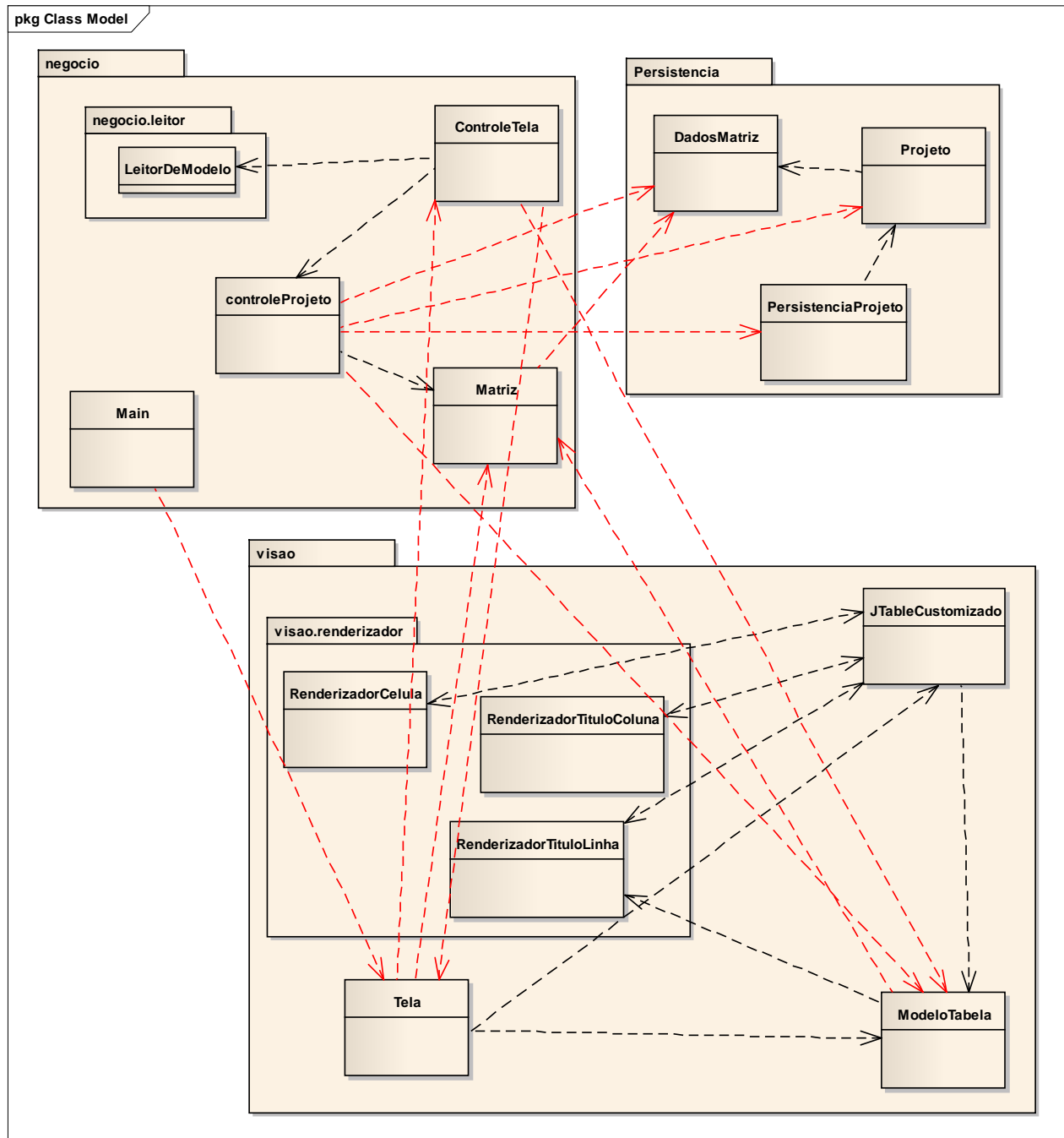
4.1 Implementasi

4.1.1 Persiapan Data

Dalam tahap implementasi ini langkah pertama yang dilakukan adalah menyiapkan data. Data yang digunakan adalah *sourcecode* dari aplikasi Trama. Didalam aplikasi Trama terdiri atas tiga paket yaitu "negocio", "persistencia" dan "visao". Dua dari tiga paket memiliki sub paket yaitu "negocio.leitor", "negocio.leitor.interface" dan "visao.renderizador". Satu buah *interface* yaitu "PluginInterface" yang ada didalam elemen sub paket "negocio.leitor.interface". Kemudian 11 kelas yang tersebar dalam paket dan sub paketnya. Tabel 4.1 menunjukkan daftar isi dari elemen-elemen yang terdapat dalam aplikasi Trama. Berdasarkan pada kolom nama paket yang terdapat pada Tabel 4.1 terlihat bahwa terdapat struktur hierarki pada paket no.2, 3 dan no.6. Di dalam sistem Trama terdapat tiga paket yang menduduki level paling atas, yaitu : "Negocio", "Persistencia" dan "Visao". Disebut dengan paket level 0. Selanjutnya didalam paket "negocio" terdapat struktur hierarki, dapat dikatakan berada pada level 1 adalah elemen-elemen yang terdapat dalam paket "negocio", yaitu sub paket "leitor" yang memiliki kelas "LeitorDeModelo", kelas "Controle Projeto", "Controle Tela", "Matriz" dan "Main". Kemudian didalam sub paket "leitor" terdapat elemen berupa sub paket *interface* yang didalamnya terdapat sebuah *interface* yaitu "Plugin Interface". Terlihat bahwa di dalam sistem Trama memiliki 14 kelas-kelas dan sebuah *interface*. Setelah mengetahui elemen-elemen apa saja yang terdapat dalam sebuah paket, selanjutnya akan dilakukan perhitungan untuk membuat matriks similaritas yang dapat digunakan untuk mengetahui tingkat kedekatan yang terjadi antar elemen-elemen yang terdapat dalam sebuah paket.

4.1.2 Hubungan Antar Elemen Dalam Paket

Apabila elemen-elemen yang terdapat dalam sistem Trama sudah diketahui, untuk selanjutnya adalah mengetahui keterkaitan antar elemen. Relasi apa yang terjadi antara elemen satu dengan yang lainnya. Untuk memudahkan membaca hubungan antara elemen yang terdapat dalam sistem Trama dapat diperlihatkan dengan menggunakan Gambar 4.1.



Gambar 4.1. Dependensi elemen paket Dalam Diagram UML

Pada gambar 4.1 memperlihatkan hubungan atau keterkaitan kelas-kelas yang terdapat dalam sistem Trama. Garis yang berwarna hitam menunjukkan hubungan antar kelas dalam satu paket. Dan garis yang berwarna merah menunjukkan hubungan antar kelas dengan paket yang berbeda. Hubungan antara kelas dengan kelas yang lainnya dalam satu paket ini yang disebut dengan kohesi. Sedangkan hubungan antara kelas satu dengan paket yang berbeda dinamakan kopleng.

Tabel 4.1 Daftar elemen Paket Sistem Trama

No.	Paket	Elemen		
		Sub Paket	Interface	Kelas
1.	Negocio	Negocio.leitor	-	ControleProjeto ControleTela Main Matriz
2.	Negocio.leitor	negocio.leitor.Interface	-	LeitorDeModelo
3.	Negocio.leitor.Interface	-	PluginInterface	-
4.	Persistencia	-	-	DadosMatriz PersistenciaProjeto Projeto
5.	Visao	Visao.renderizador	-	JtableCustomizado ModeloTabela Tela
6.	Visao.renderizador	-	-	RenderizadorCelula RenderizadorTituloColuna RenderizadorTituloLinha

Dengan melihat hubungan dependensi yang terjadi antara elemen-elemen dalam satu paket (kohesi) maupun elemen-elemen dengan yang berada dalam paket yang lainnya, hubungan ini digunakan untuk mengetahui kedekatan antara elemen. Kedekatan antar elemen diketahui dengan menggunakan ukuran similaritas. Ukuran similaritas antara dua buah elemen diketahui dengan menghitung *Coeff*

(Persamaan 7). Nilai *Coeff* berada pada rentang 0-1, 0 menandakan bahwa dua buah elemen tidak memiliki kesamaan dan dikatakan tidak dekat, dan 1 menandakan dua buah elemen memiliki kesamaan dan dekat. Tabel 4.2 merupakan tabel yang berisi kelas-kelas yang memiliki keterkaitan atau dependensi dengan kelas lainnya. Misalnya : kelas “ControleProjeto” memiliki dependensi dengan kelas “matriz”, “dadosMatriz”, “PersistenciaProjeto”, “Projeto” dan “ModeloTabela”. Kemudian kelas “ControleTela” memiliki dependensi dengan kelas “LeitorDeModelo”, “ControleProjeto”, “ModeloTabela” dan kelas “Tela”.

Tabel 4.2 Relasi Antar Kelas Dalam Sistem Trama

Kode Kelas	Nama Kelas	Anggota relasi kelas
c1	ControleProjeto	{c4, c7, c8, c9, c11}
c2	ControleTela	{c1, c5, c11, c12}
c3	Main	{c12}
c4	Matriz	{c7}
c5	LeitorDeModelo	{c5}
c6	PluginInterface	{c6}
c7	DadosMatriz	{c7}
c8	PersistenciaProjeto	{c9,}
c9	Projeto	{c7}
c10	JTableCustomizado	{c11, c13, c14, c15}
c11	ModeloTabela	{c4, c15}
c12	Tela	{c2, c4, c10, c11}
c13	RenderizadorTituloCelula	{c10}
c14	RenderizadorTituloColuna	{c10}
c15	RenderizadorTituloLinha	{c10}

Berdasarkan pada kelas yang memiliki anggota relasi sama, maka dapat ditentukan ukuran kesamaan atau similaritas antara dua kelas yang berpasangan,

dimana ukuran kesamaan ini dinyatakan dalam sebuah nilai *Coeff.* selanjutnya dilakukan perhitungan *Coeff* dari tiap kelas yang berpasangan. Nilai *coeff* ini menandakan nilai kedekatan atau kemiripan suatu kelas yang berpasangan. Misalnya: Berapa nilai kedekatan (*Coeff*) berdasarkan pada relasi dependensi antara “ControleTela” dengan “ControleProjeto”? *Coeff* antara “ControleTela” dan “ControleProjeto” diperoleh sebesar 0.125. Berdasarkan pada Tabel 4.2 dapat diketahui bahwa kelas c1 (ControleProjeto) memiliki dependensi dengan beberapa kelas yang tergabung dalam himpunan c1 anggotanya adalah {c4, c7, c8, c9, c11}. Himpunan c2 {c1, c5, c11, c12}. Berdasarkan pada data yang terdapat pada kedua himpunan tersebut maka dapat dihitung similaritas (persamaan 2) antara c1 dan c2 adalah sebanyak 0.125. diperoleh dengan rincian sebagai berikut : $Coeff(c2,c1) = \frac{1}{1+(3+4)} = 0.125$. Nilai *Coeff* semua pasangan kelas ini dituangkan dalam bentuk matriks similaritas seperti yang terdapat dalam Tabel 4.3 yang menunjukkan nilai *Coeff* kedekatan antar elemen atau kelas. Dengan mengetahui seberapa dekat keterhubungan antar kelas, dapat kita kelompokkan kelas-kelas yang memiliki kedekatan lebih akan dikelompokkan menjadi satu kelompok atau satu paket.

Teknik yang digunakan untuk mengelompokkan kelas-kelas ini adalah *clustering* dengan mencoba diterapkan pada Metode *single Linkage*. Dengan menggunakan metode *single Linkage* masing-masing kelas akan dianggap sebagai sebuah *single cluster*. Selanjutnya tiap *single cluster* akan mendekat atau mengelompok menjadi satu kelompok berdasarkan nilai kedekatannya. Nilai kedekatan antar cluster dilihat berdasarkan pada maksimum nilai *Coeff*. Apabila setiap kelas yang berpasangan memiliki nilai *Coeff* maka dapat dikatakan bahwa pasangan kelas tersebut memiliki nilai kedekatan. Sebagaimana yang telah dijelaskan pada sub bab sebelumnya bahwa rentang *Coeff* adalah 0-1. Jika setiap pasangan memiliki nilai *Coeff* mendekati nilai 1 atau mungkin mencapai rentang nilai maksimal yaitu 1 maka dianggap bahwa pasangan ini memiliki kedekatan atau kesamaan. Didalam matriks similaritas hubungan antara elemen $A \rightarrow B$ ataupun $B \rightarrow A$ memiliki nilai *coeff* yang sama, hal ini dikarenakan nilai *coeff* ini diperoleh berdasarkan pada kesamaan kelas yang diakses diantara kedua elemen yang berpasangan.

Tabel 4.3 *Coeff* Matriks Similaritas Antar Kelas Dalam Sistem Trama

kelas	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15
c1	0.000	0.125	0.000	0.200	0.000	0.000	0.200	0.200	0.200	0.125	0.167	0.286	0.000	0.000	0.000
c2	0.125	0.000	0.250	0.000	0.250	0.000	0.000	0.000	0.000	0.143	0.000	0.143	0.000	0.000	0.000
c3	0.000	0.250	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
c4	0.200	0.000	0.000	0.000	0.000	0.000	1.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
c5	0.000	0.250	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
c6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
c7	0.200	0.000	0.000	1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
c8	0.200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
c9	0.200	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
c10	0.125	0.143	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.200	0.143	0.000	0.000	0.000
c11	0.167	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.200	0.000	0.200	0.000	0.000	0.000
c12	0.286	0.143	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.143	0.200	0.000	0.250	0.250	0.250
c13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	1.000	1.000
c14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	1.000	0.000	1.000
c15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	1.000	1.000	0.000

Setelah didapatkan Matriks similaritas yang berisi nilai *Coeff*, matriks ini digunakan untuk mengelompokkan kelas-kelas menggunakan Metode *Single Linkage*. Pertama kalinya setiap kelas dianggap sebagai satu *cluster*. Nilai *Coeff* dianggap sebagai jarak antar *cluster*. Kelas akan mengelompok menjadi satu *cluster* jika jaraknya berdekatan. Tabel 4.4 menunjukkan hasil pengelompokan kelas-kelas dengan jumlah dua *cluster*. Pengelompokan dengan hasil tiga *cluster* ditunjukkan dengan Tabel 4.5.

Tabel 4.4 Tabel Hasil Pengelompokan Metode *Single Linkage* dengan 2 *cluster*

Cluster	Anggota
1	c6
2	c1, c2, c3, c4, c5, c7, c8, c9, c10, c11, c12, c13, c14, c15

Tabel 4.5 Tabel Hasil Pengelompokan Dengan Metode *Single Linkage* dengan 3 *cluster*

Cluster	Anggota
1	c2, c3, c5
2	c1, c4, c7, c8, c9, c10, c11, c12, c13, c14, c15
3	c6

Tabel 4.6 Tabel Hasil Pengelompokan Dengan Metode *Single Linkage* dengan *cutoff* 0.8

Cluster	Anggota
1	c2, c3, c5
2	c6
3	c1, c4, c7, c8, c9, c10, c11, c12, c13, c14, c15

4.2. Pengujian

Pada sub bagian ini akan membahas tentang pengujian hasil pengelompokan atau *refactoring* paket. Pengujian yang dilakukan adalah pengujian kohesi tiap paket untuk mengetahui bahwa dengan mengimplementasikan teknik *clustering* mampu meningkatkan nilai kohesi paket. Kemudian pengujian yang kedua adalah pengujian dengan nilai *variance* untuk mengetahui variansi dari *cluster* mana yang paling menghasilkan *cluster* ppaket dengan nilai kohesi yang optimal. Pengujian ini dilakukan agar dapat menjawab tujuan dari penelitian ini yaitu mendapatkan ukuran *cluster* yang optimal dengan memiliki nilai kohesi paket yang tinggi serta tidak melanggar syarat atau ketentuan dari teknik *clustering* paket yaitu : sangat tidak diharapkan menghasilkan cluster yang tidak berimbang, yaitu terdapat *single cluster* dan *cluster* besar. Bahwa *cluster* yang ideal harus memiliki ukuran yang tepat, contohnya sebuah *cluster* memiliki jumlah anggota penuh sementara satu buah *cluster* berikutnya hanya memiliki satu anggota.

4.2.1 Pengujian Kohesi Paket

Pada pengujian kohesi , yang pertama kali dilakukan adalah dengan melakukan uji nilai kohesi terhadap data uji yaitu paket yang terdapat dalam sistem Trama awal (sebelum dilakukan *refactoring*) dengan paket dari sistem Trama setelah dilakukan *refactoring*.

Tahapan pengujian kohesi paket dilakukan dengan cara menghitung nilai kohesi dari masing-masing paket. Nilai kohesi masing-masing paket akan dibandingkan dengan nilai kohesi paket sesudah dilakukan pengelompokan dengan metode *Single Linkage*. Pengujian pertama perhitungan nilai kohesi paket. Hasil perhitungan nilai kohesi dari data uji paket dalam sistem trama pada kondisi awal (sebelum *refactoring*) ditunjukkan dalam Tabel 4.7. Pada kondisi awal sistem trama terdiri atas tiga paket yaitu "negocio", "persistencia" dan "visao". Setiap paket berisi elemen-elemen dimana dapat berupa sub paket, kelas atau *interface*. Kolom elemen berisi elemen-elemen yang terdapat dalam setiap paket pada sistem trama. Kolom relasi berisi hubungan antara elemen satu dengan yang lainnya dalam sebuah paket. Kolom n adalah jumlah elemen dalam setiap paket. Sedangkan jumlah relasi ditunjukkan dalam kolom r . Nilai kohesi setiap paket terdapat pada kolom $Pcoh$, didapatkan dengan cara melakukan perhitungan menggunakan Persamaan (10). Hasil pengukuran nilai kohesi paket sebelum *refactoring* seperti yang telah ditampilkan pada Tabel 4.7 adalah untuk negocio: sebesar 0.15, persistencia = 0.33 dan visao sebesar 0.42. Ketika menghitung nilai kohesi setiap paket, sangat diperhatikan elemen-elemen yang terdapat di dalam sebuah paket. Kondisi awal paket dari sistem trama ini tidak hanya berisi kelas atau interface saja, akan tetapi terdapat beberapa paket yang memiliki sub paket. Dengan adanya sub paket ini akan mempengaruhi perhitungan relasi yang terjadi didalamnya. Jumlah relasi akan sangat berpengaruh terhadap nilai kohesi yang didapatkan. Selain itu juga jumlah elemen sangat menentukan nilai kohesi paket juga.

Perhitungan selanjutnya adalah untuk mengetahui nilai kohesi tiap paket setelah proses *refactoring* menggunakan metode *clustering single linkage*, ditunjukkan dalam Tabel 4.8.

Tabel 4.7 Nilai Kohesi Awal (Sebelum Pengelompokan)

No	Package	Elemen	Relasi	n	r	PCoh
1	negocio	subpackage leitor, class controleProjeto, class ControleTela, class Main, Class Matriz	Class ControleTela→Subpackage leitor ControleProjeto→Matriz ControleTela →ControleProjeto	5	3	0.15
2	persistencia	class DadosMatriz, class PersistenciaProjeto, class Projeto	class PersistenciaProjeto→Projeto class Projeto →DadosMatriz	3	2	0.33
3	visao	subpackage renderizador, class jTableCustomizado, class ModeloTabela, Class Tela	class jTableCustomizado →ModeloTabela class Modelo Tabela →subpackage renderizador Class Tela→jTableCustomizado class Tela→modeloTabela sub package renderizador→class jTableCustomizado	4	5	0.42

Pada tabel 4.7 terlihat bahwa paket visao memiliki nilai kohesi paling tinggi dibandingkan dengan paket-paket yang lainnya. Didalam kolom relasi terlihat didalam paket "visao" terdapat relasi *circle*, yaitu relasi bolak balik, yang terjadi antara kelas "jTableCustomizado" dengan Sub Paket "Renderizador". Paket "persistencia" menduduki urutan kedua setelah "visao" dan paket "negocio" memiliki nilai kohesi yang paling rendah dibandingkan dengan "visao" dan "persistencia".

Table 4.8 Nilai Kohesi Setelah Pengelompokan dengan *SingleLinkage* Dengan hasil 2 *cluster*.

Package	Elemen	Relasi	n	r	PCoh
1	PluginInterface	-	1	-	1
2	ControleProjeto, ControleTela, Main, Matriz, leitorDeModelo, DadosMatriz, PersistenciaProjeto, Projeto, JtableCustomizado, ModeloTabela, Tela, RenderizadorCelula, RenderizadorTituloColuna, RenderidadorTituloLinha	ControleProjeto→Matriz ControleProjeto→DadosMatriz ControleProjeto→PersistenciaProjeto ControleProjeto→Projeto ControleProjeto→ModeloTabela ControleTela→LeitorDeModelo ControleTela→ControleProjeto ControleTela→ModeloTabela ControleTela→Tela Main→Tela Matriz→DadosMatriz PersistenciaProjeto→Projeto Projeto→DadosMatriz JtableCustomizado→ RenderizadorCelula JtableCustomizado→ RenderizadorTituloColuna JtableCustomizado→ RenderidadorTituloLinha JtableCustomizado→ModeloTabela ModeloTabela→Matriz ModeloTabela→ RenderidadorTituloLinha Tela→ControleTela Tela→Matriz Tela→ JtableCustomizado Tela→ModeloTabela	14	26	0.14

Tabel 4.9 Nilai Kohesi Setelah Pengelompokan dengan *SingleLinkage* Dengan hasil 3 *cluster* maupun *cut off* 0.8

Package	Elemen	Relasi	n	r	PCoh
1	ControleTela, Main, LeitorDeModelo	ControleTela→LeitorDemodelo	3	1	0.16
2	PluginInterface	-	1	-	1
3	ControleProjeto, Matriz, DadosMatriz, PersistenciaProjeto, Projeto, JtableCustomizado, ModeloTabela, Tela, RenderizadorCelula, RenderizadorTituloColuna, RenderidadorTituloLinha	ControleProjeto→Matriz ControleProjeto→DadosMatriz ControleProjeto→PersistenciaPro jeto ControleProjeto→Projeto ControleProjeto→ModeloTabela Matriz→DadosMatriz PersistenciaProjeto→Projeto Projeto→DadosMatriz JtableCustomizado→ RenderizadorCelula JtableCustomizado→ RenderizadorTituloColuna JtableCustomizado→ RenderidadorTituloLinha JtableCustomizado→ModeloTabe la ModeloTabela→Matriz ModeloTabela→ RenderidadorTituloLinha RenderizadorCelula→JtableCusto mizado RenderizadorTituloColuna→Jtabl eCustomizado RenderidadorTituloLinha→ JtableCustomizado Tela→Matriz Tela→JtableCustomizado Tela→ModeloTabela	1 1	20	0.18

4.2.2 Pengujian variansi *cluster*

Untuk menentukan ukuran *cluster* yang ideal dapat dilakukan pengujian dengan menggunakan nilai variansi *cluster*. Nilai ini diukur dengan menghitung nilai *variance* diantara *cluster* dan *variance* didalam *cluster*. Masing-masing *cluster* dengan ukuran yang berbeda akan mendapatkan nilai V_b dan V_w . Setelah didapatkan kedua nilai ini maka untuk mendapatkan hasil akhir yaitu nilai *variance*. Hasil perhitungan nilai *variance* dari masing-masing ukuran *cluster* yang di kelompokkan dengan metode *single linkage* dengan ukuran *cluster* dua menghasilkan nilai *variance* sebesar 3, 572 dan untuk ukuran *cluster* tiga sebanyak 4,885.

4.3. Analisis dan Pembahasan

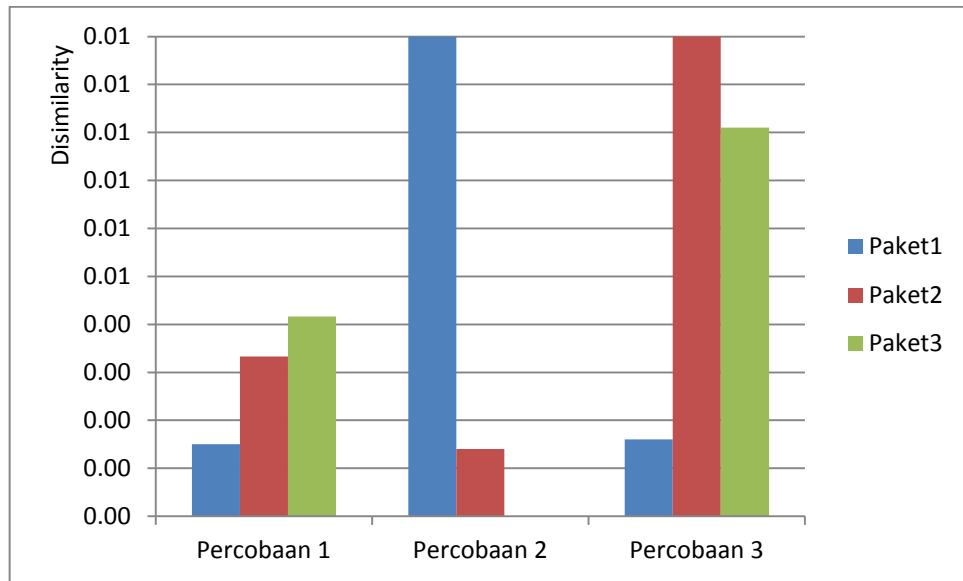
Pada sub bab ini akan dilakukan analisis terhadap hasil pengujian yang telah dilakukan. Sehingga dengan lakukan langkah analisis maka akan menjawab tujuan utama dari dilakukannya penelitian kali ini.

4.3.1. Analisis Hasil Pengujian nilai Kohesi

Berdasarkan pada hasil hasil nilai kohesi yang diperoleh dari hasil uji pada Tabel 4.7-4.9 dapat dibuatkan tabulasi untuk mengetahui peningkatan nilai kohesi yang didapatkan setelah dilakukan proses *refactoring*. hasil analisis ini dapat ditunjukkan dalam Tabel 4.10

Tabel 4. 10 Tabulasi Nilai Kohesi dari Ketiga Pengujian

Pengujian	Nama Paket	Nilai Kohesi
1	1	0.15
	2	0.33
	3	0.42
2	1	1
	2	0.14
3	1	0.16
	2	1
	3	0.81



Gambar 4.2. Diagram Batang Menunjukkan Perbandingan Nilai Kohesi

Berdasarkan pada hasil tabulasi yang diperoleh seperti yang telah di tuangkan dalam Tabel 4.10 merupakan hasil pengukuran nilai kohesi dari tiap paket pada setiap pengujian. Terlihat bahwa pada pengujian 2 dan 3 nilai kohesi dari paket meningkat. Pada percobaan 1 paket 1 memperoleh nilai kohesi 0.15 dan pada percobaan 2 paket 1 nilai kohesinya meningkat tajam dengan selisih 0.85. Dengan kenaikan yang tajam ini tentunya sangat sesuai dengan tujuan penelitian yaitu meningkatkan nilai kohesi. Akan tetapi kenaikan nilai kohesi ini tidak diimbangi dengan penyebaran kelas-kelas didalam paket. Pada percobaan ke-2 terlihat munculnya penyebaran kelas yang tidak seimbang. Terdapat *single cluster* yang berisi hanya 1 kelas. Dan *cluster* lainnya adalah cluster besar yang berisi 14 kelas. Tentunya keadaan ini tidak diharapkan.

Pada percobaan yang ke-3 terlihat bahwa dari ketiga paket yang dihasilkan mengalami peningkatan. Paket 1 meningkat sebesar 0.1, paket 2 sebesar 0.67 dan paket 3 sebesar 0.39. berdasarkan nilai kohesi yang diperoleh menunjukkan mengalami peningkatan dibandingkan dengan nilai kohesi paket pada percobaan 1. Selain itu juga tidak terdapat *single cluster* dan *cluster* besar.

Untuk lebih memudahkan melihat hasil nilai kohesi paket pada percobaan 1, 2 dan 3 dapat di lihat pada diagram batang seperti yang terdapat pada Gambar 4.2. Nilai kohesi pada paket setelah dicoba untuk melakukan perpindahan kelas ke paket lainnya ternyata menghasilkan nilai kohesi yang lebih tinggi dibandingkan dengan nilai kohesi pada kondisi paket yang semula dari sistem Trama. Percobaan dilakukan dengan memindahkan kelas-kelas yang memiliki hubungan atau relasi dengan kelas lain digabungkan menjadi satu paket. Kelas-kelas yang ada dicoba dikelompokkan dengan menggunakan metode Single Linkage. Hasil nilai kohesi paket kondisi awal (sebelum dikelompokkan) jika dibandingkan dengan nilai kohesi paket setelah dikelompokkan dengan metode Single Linkage mengalami peningkatan.

Berdasarkan percobaan yang dilakukan pada pengujian nilai kohesi ini sudah mampu digunakan untuk menjawab bahwa proses *refactoring* dengan menggunakan metode *clustering* mampu meningkatkan nilai kohesi paket.

Nilai kohesi pada suatu paket diperoleh dengan menggunakan perhitungan pada persamaan 10. Dimana nilai kohesi ini diperoleh dipengaruhi oleh dua hal yaitu elemen yang terdapat dalam paket dan relasi yang terjalin diantara elemen-elemen yang terdapat di dalam paket tersebut. Relasi yang terjadi antara elemen paket misalnya kelas-kelas, kelas-interface, kelas-sub paket ataupun sub paket-kelas sangat memiliki pengaruh yang penting dalam menentukan nilai kohesi. Misalnya jika terdapat suatu kondisi dalam sebuah paket memiliki elemen sebanyak $n > 1$, tetapi tidak terdapat relasi didalamnya, maka nilai kohesi yang diperoleh =0.

Selain itu hubungan atau relasi bolak balik atau yang disebut dengan *cyclic* dependensi antara elemen paket juga sangat mempengaruhi nilai kohesi. Dan apabila terdapat hubungan atau relasi bolak balik yaang terjadi antara elemen-elemen agar seharusnya elemen-elemen ini ditempatkan dalam satu paket. Pada sistem Trama ini terjadi *cyclic* dependensi terjadi pada kelas "JtabelCustomizado" dengan Sub Paket "Randerizador". Kedua kelas ini berada pada paket "Visao", dan juga terjadi antara kelas "ControleTela" dengan kelas "Tela".

4.3.2. Analisis Hasil Pengujian Variansi Cluster

Berdasarkan pada hasil pengujian dengan menggunakan perhitungan nilai variansi pada hasil *cluster* yaitu pada hasil cluster yang dihasilkan pada percobaan 2 dengan menghasilkan ukuran *cluster* 2 menghasilkan nilai *variance* sebesar 3,572 dan untuk ukuran *cluster* 3 sebanyak 4,885. Nilai *variance* pada ukuran *cluster* 3 lebih tinggi dibandingkan dengan *cluster* 2 menunjukkan bahwa ukuran *cluster* 3 lebih optimal dibandingkan dengan cluster 2. Hal ini sejalan dengan hasil yang diperoleh jika dilihat dengan nilai kohesi yang didapatkan. Serta tidak bertentangan dengan tujuan clustering itu sendiri yaitu mendapatkan hasil *cluster* dengan komposisi anggota *cluster* yang berimbang yaitu tidak menemui *single cluster* dan *cluster besar*.

Sebuah *cluster* seharusnya didesain dengan ukuran yang seimbang. Tujuan dari *clustering* adalah pengelompokkan suatu data menjadi lebih modular. Sangat dihindari apabila terjadi ketidakseimbangan atau ukuran *cluster* yang ekstrim. Ukuran *cluster* yang dihasilkan pada percobaan 2 merupakan contoh dari ketidakseimbangan tersebut. Terbukti dengan hasil paket 1 berisi *single cluster* dan paket 2 berisi *cluster* besar. Sehingga tujuan dari teknik *clustering* paket yang salah satunya adalah untuk menghindari adanya *cluster* tunggal dan *cluster* besar tidak dapat terpenuhi.

Metode *single linkage* sebenarnya sangat rawan untuk dapat menghasilkan *single cluster*. Karena sifat dari metode ini adalah menentukan setiap objek sebagai *cluster* dan kemudian antara objek yang satu dengan yang lainnya baru dihitung jarak terdekatnya. Apabila terdapat sebuah objek dengan nilai jarak terdekat yang terlalu jauh dengan tetangganya akan menyebabkan objek ini tidak digabungkan dengan *cluster* lainnya. Oleh sebab itu dalam penelitian kali ini dicari paket mana yang mampu menghasilkan nilai kohesi tinggi tetapi juga setiap *cluster* mempunyai penyebaran anggota yang berimbang dengan *cluster* lainnya.

[Halaman ini sengaja dikosongkan]

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil berdasarkan pada uji coba dan analisis hasil uji coba yang dilakukan dalam penelitian ini adalah sebagai berikut.

1. Berdasarkan pada hasil refactoring yang dilakukan, terbukti bahwa nilai kohesi yang dihasilkan dengan melakukan proses refactoring dengan metode *single linkage* terbukti dapat meningkatkan nilai kohesi paket. Hal ini dipengaruhi oleh adanya perpindahan kelas-kelas yang memiliki hubungan atau relasi dengan kelas yang lainnya akan dikelompokkan dalam satu paket. Sehingga dapat dikatakan bahwa besarnya nilai kohesi pada sebuah paket bergantung pada elemen yang terdapat di dalam paket dan relasi yang terjalin diantara elemen-elemen di dalamnya.
2. Nilai kohesi yang paling optimal didapatkan dari ukuran *cluster* paling optimal yang dihasilkan oleh metode *single Linkage* dengan data uji sistem trama adalah 3 cluster berdasarkan pada *cutoff* 0.8. ukuran cluster 3 dianggap paling optimal dengan tetap memperhatikan sebuah pendekatan dari teknik *clustering* itu sendiri bawah sebuah *cluster* yang baik tidak disarankan menghasilkan jumlah *cluster* yang ekstrem yaitu menghasilkan *single cluster* dan yang lainnya menghasilkan satu buah *cluster* besar karena akan menyebabkan ketidakseimbangan anggota *cluster*.

5.2 Saran

Adapun saran yang bisa diberikan berdasarkan hasil yang didapat dari penelitian ini adalah sebagai berikut:

1. Hubungan kedekatan antara elemen yang terdapat dalam paket (kelas, *interface*, maupun sub paket) dapat digali lebih dalam dengan menggunakan beberapa atribut yang dapat digunakan untuk mengukur

kedekatan misalnya: *instance of class*, *type parameter* dan *variable*. Tiap-tiap atribut yang digunakan untuk mengetahui kedekatan ini diberikan bobot sehingga dapat diketahui seberapa besar tingkat kepentingannya didalam sebuah elemen. Sehingga dapat diketahui mana atribut yang paling mendukung dan mana yang kurang mendukung terhadap hubungan kedekatan antar elemen.

2. Metode *clustering* yang digunakan pada penelitian kali ini memiliki kelemahan yaitu sangat rawan untuk menghasilkan *cluster* yang tidak berimbang. Sehingga pada penelitian selanjutnya agar dapat dilakukan dengan mencoba menggunakan metode *clustering* lainnya seperti *complete link* ataupun *average link*. Dengan begitu akan didapatkan hasil perolehan nilai kohesi yang paling optimal jika dibandingkan dengan metode *clustering* yang lainnya.

DAFTAR PUSTAKA

- [1] Martin Fowler, *Refactoring: Improving the Design of Existing Code*. Boston: Addison-Wesley Longman Publishing Co. Inc., 1999
- [2] A. Ananda Rao and K. Narendar Reddy. 2011. *Identifying Clusters of Concepts in a Low Cohesive Class for Extract Class Refactoring Using Metrics Supplemented Agglomerative Clustering Technique*. International Journal of Computer Science Issues, Vol. 8, Issue 5, No 2
- [3] P. Gupta, G. Kaur. 2015. "A Concept of A-KNN Clustering in Software Engineering". International Journal of Computer Applications (0975 – 8887) Volume 119 – No.18.
- [4] A. Alkhalid, M. Alshayeb, and S.A. Mahmoud, "Software refactoring at the function level using new adaptive K-Nearest Neighbor Algorithm" .Advances in Engineering Software 41(2010) 1160 – 1178.
- [5] A. Alkhalid, M. Alshayeb, and S.A. Mahmoud, "Software refactoring at the package level using clustering techniques," IET Software, vol. 5, no. 3, pp. 274-286, 2011.
- [6] C. Srinivasa, V.Radhakrishnab dan C.V. Guru Rao. 2014. *Clustering software components for program restructuring and component reuse using hybrid XNOR similarity function*. Procedia Technology 12 (2014) 246 – 254
- [7] S. Singh, K. Kaur. 2015."Enhancement in A-KNN Clustering Technique To Analyse Software Architecture". International Journal of Computer Science and Communication Engineering. Volume 4 issue 1.
- [8] Ishu, S. Singh, 2015. " To Enhance A-KNN Clustering Algorithm for Improving Software Architecture". International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 3949-3954
- [9] Lung, C.-H., Xu, X., Zaman, M., Srinivasan, A.: „Program restructuring using clustering techniques“, J. Syst. Softw., 2006, 79, (9), pp. 1261–1279
- [10] Lung, C.-H., Zaman, M., Nandi, A.: „Applications of clustering techniques to software partitioning, recovery and restructuring“, J. Syst. Softw., 2004, 73, (2), pp. 227–244

- [11] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, “Resource ReSerVation Protocol” (RSVP), 1997.
- [12] Fabio, M.: Trama Source Forge, <http://sourceforge.net/projects/trama/>. Accessed: 22 November 2008
- [13] Shrestha, S., Gurung, Y.: Front End For MySQL Source Forge, <http://sourceforge.net/projects/frontend4mysql>. Accessed: 16 November 2008
- [14] Gupta V, Chhabra JK. Package coupling measurement in object-oriented software. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 24(2): 273{283 Mar. 2009
- [15] Gupta V, Chhabra JK, Package level cohesion measurement in object-oriented software. J Braz Comput Soc (2012) 18:251–266.
- [16] P.I. Beni, dkk, Implementasi Metode Single Linkage Untuk menentukan Kinerja Agent Pada Call Center Berbasis Asteriks For Java, PENS ITS.

