Jurnal Sains, Teknologi dan Industri, Vol. 12, No. 1, Desember 2014, pp. 136 - 145
ISSN 1693-2390 print/ISSN 2407-0939 online

PENERAPAN *HIERARCHICAL CLUSTERING* PADA PENGUKURAN KOHESI KELAS

Inggih Permana[1], Zarnelly[2]
[12]Universitas Islam Negeri Sultan Syarif Kasim Riau
[1]inggihpermana@uin-suska.ac.id, [2]zar_nelly@yahoo.com

ABSTRAK

Pengukuran kohesi pada sebuah kelas merupakan parameter penting dalam pengukuran kualitas sebuah perangkat lunak. Telah banyak penelitian dilakukan untuk hal ini, salah satunya menggunakan *hierarchical clustering*. Teknik pengukuran kohesi menggunakan *hierarchical clustering* yang ditawarkan oleh penelitian sebelumnya tidak mempertimbangkan hubungan tidak langsung antara *method* dan *instance* variabel. Studi ini bertujuan untuk memperbaiki hal tersebut dengan cara memodifikasi matriks EP sehingga matriks tersebut mempertimbangkan hubungan tidak langsung antara *method* dan *instance* variabel. Hasil percobaan menunjukkan bahwa dengan menggunakan teknik ini tidak hanya dapat meningkatkan kemampuan pengukuran kohesi, tetapi juga dapat membagi *method-method* dan *instance* variabel-variabel dalam beberapa *cluster* menurut fungsi dan penggunaan umumnya

Kata Kunci : Kelas, Kohesi, *Hierarchical Clustering*, Matriks EP, Matriks EP yang dimodifikasi, *Method*, *Instance* Variabel

ABSTRACT

Measurement of cohesion in a class is an important parameter in measuring the quality of a software. There have been many studies done on this subject, one of them using hierarchical clustering. Cohesion measurement technique using hierarchical clustering offered by previous studies did not consider indirect relations between the methods and instance variables. This study aims to rectify this by modifying the EP matrix so that the matrix is considered an indirect relationship between the methods and instance variables. The experimental results show that by using this technique not only can improve the cohesion measurement, but also can divide those methods and instance variables into several clusters according to function and general use.

Key Words : *Class, Cohesion, Hierarchical Clustering, Matrix EP, Instance Variabel*

PENDAHULUAN

Pengukuran kohesi pada sebuah kelas telah menjadi bagian penting dalam memprediksi kualitas sebuah perangkat lunak yang dibuat dengan konsep berorientasi objek. Ini dikarenakan kohesi pada kelas menggambarkan keterhubungan antara komponen-komponen yang ada di dalam kelas. Semakin tinggi kohesi sebuah kelas maka semakin tinggi hubungan antara anggota-anggota di dalam kelas dan semakin jelas juga fungsi kelas tersebut. Sebaliknya, semakin rendah kohesi sebuah kelas maka semakin rendah hubungan antara anggota-anggota di dalam kelas dan semakin semakin rendah juga kejelasan fungsi kelas tersebut.

Banyak metode untuk mengukur kohesi pada sebuah kelas yang telah ditawarkan oleh berbagai peneliti [1], [2], [3], [4], [5], [6], [7]. Metode terbaru dikemukakan oleh L. Sadoui, M. Badri dan L. Badri[8] pada tahun 2012. Mereka menawarkan penggunaan metode *hierarchical clustering* untuk mengukur hubungan antara *method-method* dan *instance*variabel-variabel pada sebuah kelas.Hasilnya adalah didapat sebuah nilai yang lebih merefleksikan kohesi dan juga struktur sebuah kelas.

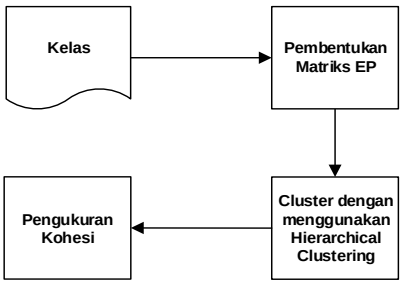
Penelitian ini bertujuan untuk meningkatkan kemampuan metode pengukuran kohesi pada sebuah kelas yang dilakukan oleh L. Sadoui, M. Badri dan L. Badri [8] sebelumnya.Penelitian mereka tidak mempertimbangkan hubungan tidak langsung

antara *method* dan *instance* variabel. Penelitian ini juga menggunakan dua buah metode pengukuran kohesi terkenal, yaituLCOM5 [4] dan TCC [6] sebagai perbandingan.

Makalah ini dibagi menjadi beberapa bagian untuk mempermudah mengemukakan ide yang akan kami tawarkan. Bagian 1, bagian ini, merupakan bagian yang membahas tentang latar belakang dan tujuan penelitian yang dilakukan. Bagian 2 menjelaskan tentang desain yang digunakan dalam penelitian ini.Bagian 3 menjelaskan sekilas tentang teori-teori dasar kohesi pada class. Bagian 4 akan menjelaskan sekilas tentang metode *cluster* yang akan digunakan pada penelitian ini, yaitu *hierarchical clustering*. Bagian 5 menjelaskan tentang metode yang ditawarkan oleh Lazhar, Mourad dan Linda [8], sebagai ide dasar penelitian ini. Bagian 6 menjelaskan tentang ide yang ditawarkan oleh penelitian ini, yaitu penambahan hubungan tidak langsung antara *method* dan *instance* variabel sebelum *cluster* dilakukan. Bagian 7 adalah percobaan dan analisis terhadap metode yang ditawarkan penelitian ini. Bagian 8 merupakan bagian terakhir dari makalah ini, berisi tentang kesimpulan dan saran kedepan.

DESAIN KERJA

Gambar 1 merupakan desain secara umum teknik pengukuran kohesi yang dilakukan dalam penelitian ini.Desain ini dibuat untuk mempermudah memahami bagaimana sebenarnya teknik yang ditawarkan dalam studi inibekerja.



Gambar 1 Desain umum penelitian

Kelas yang digunakan dalam penelitian ini berjumlah dua buah. Kelas pertama berasal dari penelitian St. Hanenberg dan R. Unland [12] dan kelas kedua berasal dari penelitian M. J. T. Monteiro [13]. Kedua kelas tersebut juga digunakan dalam penelitian pengukuran kohesi dengan menggunakan *clustering* sebelumnya [8]. Dari kedua kelas tersebut akan dibuat matriks EP, Matriks ini akan dijelaskan pada bagian 5 (matriks EP berdasarkan penelitian sebelumnya [8]) dan 6 (matriks EP yang ditawarkan oleh penelitian ini). Matriks EP tersebut kemudian menjadi sumber data untuk proses peng-*cluster*-an dengan menggunakan metode *hierarchical clustering*. Metode ini akan dijelaskan secara singkat pada bagian 4. Peng-*cluster*-an dilakukan dengan *tool* statistik yang bernama XLSTAT, *tool* ini dapat diunduh di http://www.xlstat.com. Hasil*cluster*akan digunakan untuk pengukuran secara manual nilai kohesi. Selanjutnya dilakukan analisis terhadap hasil perhitungan kohesi yang didapat.

KOHESI PADA KELAS

Kohesi adalah ukuran seberapa baik baris-baris kode sumber (*source code*) bekerja sama di dalam sebuah modul untuk menyediakan fungsionalitas yang spesifik [9]. Sebuah kelas memiliki kohesif yang baik apabila *method-method* yang ada di dalam kelas tersebut memiliki keterhubungan yang dekat tetapi tetap melakukan fungsi tunggal dikelas tersebut.

Keterhubungan antar *method* tersebut bisa dikarenakan oleh *method-method* tersebut menggunakan *instance* variabel yang sama atau bisa juga karena sebuah *method* memanggil *method* lain di tubuhnya. Perlu diketahui *method-method* dan *instance* variable-variabel warisan dari kelas *parent* tidak digunakan dalam perhitungan kohesi [9].

Tabel 1 merupakan *roadmap* perkembangan metode perhitungan kohesi pada kelas yang dibuat oleh H.S. Chae dan Y.R. Kwon [10] ditambah dengan perkembangan terbaru tentang perhitungan kohesi dari tahun 2012.

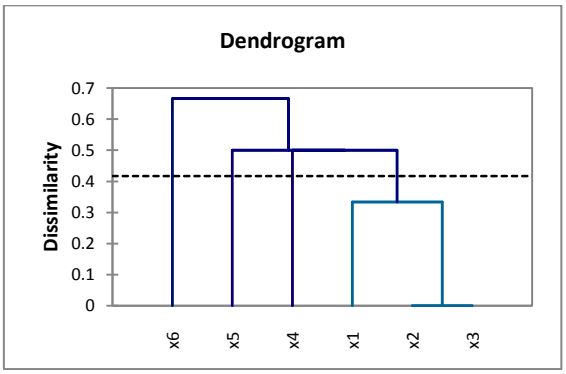
Jurnal Sains, Teknologi dan Industri, Vol. 12, No. 1, Desember 2014, pp. 136 - 145
ISSN 1693-2390 print/ISSN 2407-0939 online

TABEL 1 ROADMAP PERKEMBANGAN METODE PENGUKURAN KOHESI PADA KELAS

Nama	Tahun	Cara pengukuran
LCOM1 [1]	1991	Jumlah *method-method* berpasangan yang tidak menggunakan*instance* variabel yang sama.
LCOM2 [2]	1994	Disediakan kelas dengan *n*method$\{M_1\}$, $\{M_2\}$, ... $\{M_n\}$. Disediakan$\{I_i\}$ = himpunan *instance* variabel yang digunakan oleh *method*$\{M_i\}$. Maka ada *n* buah himpunan $\{I_i\}$, ... $\{I_n\}$. Disediakan $p = \{(I_i, I_j) I_i \cap I_j = \emptyset\}$dan $P = \{(I_i, I_j) I_i \cap I_j \neq \emptyset\}$. Jika semua n himpunan $\{I_i\}$, ... $\{I_n\}$ adalah $\emptyset$maka $P = \emptyset$. $$LCOM2 = \begin{cases} \|P\| - \|Q\|, & jika\ \|P\| > \|Q\| \\ 0, & untuk\ yang\ lainnya \end{cases}$$
LCOM3 [3]	1995	Disediakan graf yang tidak mempunyai arah G, dimana simpul-simpulnya adalah *method-method* sebuah kelas.Sisi antara dua buah simpul ada jika setidak-tidaknya dua buah *method* menggunakan satu *instance* variabel yang sama.*LCOM3* = |jumlah komponen yang terhubung dari G|
LCOM4 [3]	1995	Hampir sama dengan LCOM3, hanya saja hubungan dua buah simpul selain dari *instance* variabel,bisa juga terjadi karena sebuah *method* memanggil *method* lain di tubuhnya.
Co [3]	1995	Disediakan graf yang tidak mempunyai arah *G*, dengan simpul *V* dan sisi *E*. Maka kohesi dihitung dengan rumus $Co = 2 \times \frac{\|E\| - (\|V\| - 1)}{(\|V\| - 1) \times (\|V\| - 2)}$
LCC [4]	1995	Disediakan sebuah kelas dengan *N* buah *method* publik. Anggap *NP* sebagai jumlah maksimum *method* publik yang berpasangan. $NP = \frac{N \times (N-1)}{2}$, untuk *Nmethod* publik. Disediakan *NIC* sebagai hubungan langsung maupun tidak langsung antara *method* publik. Maka *LCC* didefinisikan sebagai jumlah relatif dari hubungan langsung maupun tidak langsung antara *method* publik. $LCC = \frac{NIC}{NP}$
TCC [4]	1995	Hampir sama dengan LCC, hanya saja TCC tidak mempertimbangkan hubungan tidak langsung antara *method* publik, atau disebut dengan NDC. Maka *TCC* didefinisikan sebagai jumlah relatif dari hubungan langsung antara *method* publik. $TCC = \frac{NDC}{NP}$
LCOM5 [5]	1996	Disediakan $\{M_i\}(1 \leq i \leq m)$, $\{A_j\}(1 \leq j \leq a)$ dan $\mu(A_i)$ sebagai himpunan *method*, himpunan *instance* variabel dan referensi *instance* variabel yang digunakan sebuah *method*. $$LCOM5 = \frac{\frac{1}{a}\sum_{1 \leq j \leq a} \|\mu(A_j)\| - m}{1 - m}$$
Coh [6]	1998	Merupakan pengembangan dari LCOM5, $Coh = \frac{\sum_{1 \leq j \leq a} \|\mu(A_j)\|}{a \times m}$
CBMC [7]	2000	Disediakan F_c sebagai faktor konektifitas dan F_s sebagai faktor struktur. Maka $CBMC = F_c \times F_s$
COH_{CL} [8]	**2012**	**Disediakan kelas dengan *n*method$\{M_1\}$, $\{M_2\}$, ... $\{M_n\}$, G_c merupakan graf hasil *clustering* dan E_A adalah simpul yang masih berhubungan setelah *n* buah *method* di-*clustering*.** **Jumlah maksimum *method* yang berpasangan adalah $\frac{N \times (N-1)}{2}$** **Maka, $COH_{CL} = \frac{\|E_A\|}{\frac{N \times (N-1)}{2}}$**

HIERARCHICAL CLUSTERING

Hierarchical clustering merupakan teknik *clustering*yang biasa ditampilkan dalam dendogram. Arah*hierarchical clustering*yang digunakan pada penelitian ini adalah *Agglomerative* yaitu dari *Ncluster* ke *kcluster*, dimana *N* adalah jumlah data dan k adalah jumlah *cluster*.



Gambar 2. Contoh dendogram hasil *hierarchical clustering*

Pada penelitian ini digunakan teknik *complete linkage* pada *hierarchical clustering*, berikut algoritmanya [11] :

1) Pada awal diasumsikan setiap data adalah *cluster*, dimana *N* jumlah data dan *k* adalah jumlah *cluster*. Berarti untuk awal *N* = *k*.
2) Hitung jarak antar cluster, pada penelitian ini digunakan metode*jaccard*.
3) Cari dua buah *cluster* yang mempunyai jarak paling maksimal, lalu gabungkan dalam *cluster* baru, sehingga *k* = *k*-1.
4) Kembali ke langkah ketiga, lakukan sampai dicapai *cluster* yang diinginkan.

PENGUKURAN KOHESI PADA MENGGUNAKAN *HIERARCHICAL CLUSTERING*

Bagian ini akan menjelaskan secara singkat tentang teknik yang dilakukan pada penelitian sebelumnya[8]. Teknik yang menjadi dasar penelitian ini. Sedangkan ide baru yang ditawarkan oleh peneltian ini akan dijelaskan dibagian 6.

Langkah awal yang dilakukan untuk pengukuran kohesi sebuah kelas dengan metode *hierarchical clustering* adalah dengan membuat matriks EP (*Entities-Properties*). Matriks EP adalah sebuah matrik yang merepresentasikan hubungan antara *method-metthod* dan *instance* variabel-variabel yang ada pada sebuah kelas[8].Disediakan sebuah kelas *C* yang mempunyai himpunan *methodM* dan himpunan *instance* variable *I*. *M* mempunyai *n* buah *method*, sehingga $M = \{m_1, m_2, ..., m_n\}$. *I* mempunya *k* buah *instance* variabel, sehingga $I = \{i_1, i_2, ..., i_n\}$. Ukuran matriks EP adalah $(n+k)\times(n+k)$, atau dengan kata lain Matriks EP merupakan matriks persegi. Matriks EP merupakan matriks biner. Jika *methodm*$_i$ menggunakan secara langsung *instance* variabel a_j, maka nilai $EP(m_i,a_j)$ dan nilai $EP(a_j,m_i)$ diisi dengan 1. Jika sebuah *methodm*$_i$ memanggil secara langsung *method* m_k maka nilai $EP(m_i,m_k)$ diisi 1.

Sebagai contoh diberikan sebuah kelas *C*yang memiliki himpunan *methodM* = $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$ dan himpunan *instance variabelI* = $\{i_1, i_2, i_3, i_4, i_5\}$.$U_i$ adalah himpunan *method*dan himpunan *instance variabel* yang digunakan oleh *method* m_i. Diberikan $U_1 = \{i_1,i_2,i_3,m_7\}$, $U_2 = \{i_1,i_2, m_7\}$, $U_3 = \{i_2,i_3\}$, $U_4 = \{i_1\}$, $U_5 = \{i_1,i_4,i_5\}$, $U_6 = \{i_4,i_5\}$ dan $U_7 = \{i_4\}$.**Tabel 2** menunjukan matriks EP hasil dari contoh ini.

Matriks EP yang didapat akan digunakan sebagai *input* dalam proses *hierarchicalclustering*. Seperti yang telah disebutkan di bagian 2 bahwa penelitian ini akan menggunakan *tool* XLSTAT. *Tool* ini juga digunakan pada penelitian sebelumnya. **Gambar 3** merupakan hasil proses *hierarchical clustering* **Tabel 2** dengan menggunakan XLSTAT.

TABEL2 CONTOH MATRIKS EP

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	i_1	i_2	i_3	i_4	i_5
m_1	0	0	0	0	0	0	1	1	1	1	0	0
m_2	0	0	0	0	0	0	1	1	1	0	0	0
m_3	0	0	0	0	0	0	0	1	1	0	0	0
m_4	0	0	0	0	0	0	0	1	0	0	0	0
m_5	0	0	0	0	0	0	0	1	0	0	1	1
m_6	0	0	0	0	0	0	0	0	0	0	1	1
m_7	0	0	0	0	0	0	0	0	0	0	1	0
i_1	1	1	1	1	1	0	0	0	0	0	0	0

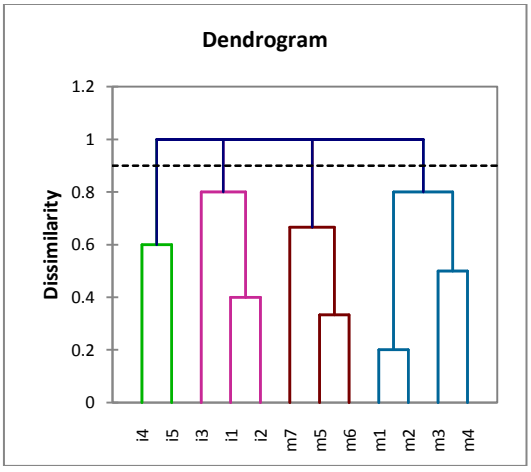
i_2	1	1	1	0	0	0	0	0	0	0	0	0
i_3	1	0	0	0	0	0	0	0	0	0	0	0
i_4	0	0	0	0	1	1	1	0	0	0	0	0
i_5	0	0	0	0	1	1	0	0	0	0	0	0

Hasil *clustering* terlihat seperti di dendogram yang terlihat pada **Gambar 3**. Di gambar tersebut terdapat 4 buah *cluster*. Untuk pengukuran kohesi yang diambil hanyalah *cluster* yang mengandung *method* saja. *Method-method* di dalam sebuah *cluster*C_i menjadi sebuah simpul yang dihubungkan satu sama lain dengan menggunakan sisi yang terdapat di himpunan E_A, sehingga terbentuk sebuah graf tidak berarah G_C seperti yang terlihat pada **Gambar 4**.

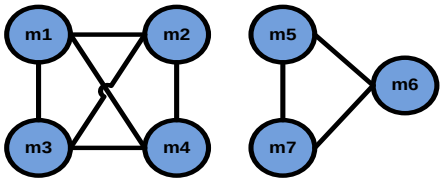
Untuk pengukuran kohesi maka harus dihitung terlebih dahulu berapa kemungkinan jumlah sisi yang terbentuk jika semua *method* yang ada dalam sebuah kelas terhubung. Perhitungan tersebut dilakukan dengan rumus $NP = \frac{N\times(N-1)}{2}$, dimana N adalah jumlah *method* [4]. Setelah itu baru dihitung sisi yang terbentuk setelah proses *clustering* dilakukan, jumlah sisi yang terbentuk itu disimbolkan dengan $|E_A|$, sehingga rumus untuk mencari kohesi sebuah kelas menjadi seperti berikut ini [8].

$$COH_{CL} = \frac{|E_A|}{\frac{N\times(N-1)}{2}} \in [0,1]$$

Pada contoh ini didapat NP = 21 dan $|E_A|$ = 9, sehingga COH_{CL} = 9/21 = 0.43.



Gambar 3. Contoh hasil peng-*cluster*-an matriks EP pada tabel 2 dengan menggunakan*hierarchical clustering*



Gambar 4. Graf hasil *clustering*.

PENGUKURAN KOHESI DENGAN MEMPERTIMBANGKAN HUBUNGAN TIDAK LANGSUNG ANTARA *METHOD* DAN VARIABEL

Bagian ini akan menjelaskantentang teknik yang ditawarkan oleh penelitian ini. Pada dasarnya ide yang ditawarkan penelitian ini hanyalah memodifikasi matriks EP sehingga memuat hubungan tak langsung antara *method* dan *instance* variabel. Misalkan sebuah *method*$m_i$memanggil secara langsung *method* m_j dan m_j menggunakan *instance* variabel i_k, maka tidak hanya $EP(m_j,m_k)$ saja yang diisi dengan nilai 1, tetapi$EP(m_j,i_k)$ juga diisi dengan nilai 1, karena m_j menggunakan i_k secara tidak langsung melalui m_j. Untuk aturan pengisian matriks EP yang lainnya sama dengan yang ada di bagian 5. Pada contoh di bagian 5 diberikan sebuah kelas C yang memiliki himpunan *method*$M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$ dan himpunan *instance* variabel$I = \{i_1, i_2, i_3, i_4, i_5\}$. U_i adalah himpunan *method* dan himpunan *instance* variabelyang digunakan oleh *method* m_i. Diberikan $U_1 = \{i_1, i_2, i_3, m_7\}$, $U_2 = \{i_1, i_2, m_7\}$, $U_3 = \{i_2, i_3\}$, $U_4 = \{i_1\}$, $U_5 = \{i_1, i_4, i_5\}$, $U_6 = \{i_4, i_5\}$ dan $U_7 = \{i_4\}$. Pada U_1 dan U_2, himpunan yang mewakili*method* dan *instance* variabelm_1 dan m_2, terlihat bahwa keduanya menggunakan *method* lain yaitu m_7. Sedangkan U_7, himpunan yang mewakili *method* dan *instance* variabel yang digunakan oleh m_7 memiliki anggotai_4. Jadi secara tidak langsung m_1 dan m_2 menggunakani_4, sehingga $U_1 = \{i_1, i_2, i_3, m_7, \boldsymbol{i_4}\}$ dan $U_2 = \{i_1, i_2, m_7, \boldsymbol{i_4}\}$. Hasil modifikasi matriks EP ini dapat dilihat di **Tabel 3**.

TABEL 3 HASIL MODIFIKASI MATRIKS EP.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	i_1	i_2	i_3	i_4	i_5
m_1	0	0	0	0	0	0	1	1	1	1	1	0
m_2	0	0	0	0	0	0	1	1	1	0	1	0
m_3	0	0	0	0	0	0	0	1	1	0	0	0
m_4	0	0	0	0	0	0	0	1	0	0	0	0
m_5	0	0	0	0	0	0	0	1	0	0	1	1
m_6	0	0	0	0	0	0	0	0	0	0	1	1
m_7	0	0	0	0	0	0	0	0	0	0	1	0
i_1	1	1	1	1	1	0	0	0	0	0	0	0
i_2	1	1	1	0	0	0	0	0	0	0	0	0
i_3	1	0	0	0	0	0	0	0	0	0	0	0
i_4	1	1	0	0	1	1	1	0	0	0	0	0
i_5	0	0	0	0	1	1	0	0	0	0	0	0

Proses selanjutnya adalah sama dengan proses yang ada di bagian 5. Matriks EP di **Tabel 3** di *clustering* dengan menggunakan *hierarchicalclustering*, lalu dibuat graf hasil *clustering*, dan yang terakhir dihitung kohesinya.

PERCOBAAN DAN ANALISIS

Percobaan ini akan menggunakan kelas yang juga telah digunakan pada penelitian tentang pengukuran kohesi dengan menggunakan *hierarchical clustering* [8] sebelumnya. Kelas *Point* berasal dari penelitianSt. Hanenberg dan R. Unland[12] dan kelas *TangleStack* berasal dari penelitian M. J. T. Monteiro[13].

Seperti yang telah dijelaskan pada bagian sebelumnya, proses *hierarchical clustering* akan menggunakan *tool* XLSTAT. Teknik *hierarchical clustering* yang digunakan adalah *complete linkage* dan perhitungan jarak antar *cluster* menggunakan *jaccard*.

A. Percobaan pada Kelas Point

Pada percobaan ini akan diukur kohesi pada kelas *point*. Seperti yang terlihat pada **Gambar 5**, kelas *point* terdiri dari 3 buah *instance* variabel (*x*, *y* dan *observers*) dan 6 buah *method* (*setX*(), *setY*(), *setXY*(), *attachObservers*(), *detachObservers*() dan *informObservers*()). Berikut*instance* variabel atau*method* yang digunakanoleh masing-masing*method* di kelas *point*:*setX*() = {*x*, *informObservers*()}, *setY*() = {*y*, *informObservers*()}, *setXY*() = {*x*, *y*, *informObservers*()}, *attachObservers*() = {*observers*}, *detachObservers*() = {*observers*} dan *informObservers*() = {*observers*}. Karena *methodsetX*(), *setY*() dan *setXY*() memanggil *methodinformObservers*() dan*informObservers*() sendiri menggunakan *instance* variabel *observers*, maka ketiga properti *method* tersebut menjadi: *setX*() = {*x*, *informObservers*(), ***observers***}, *setY*() = {*y*, *informObservers*(), ***observers***}, *setXY*() = {*x*, *y*, *informObservers*(), ***observers***}. Matriks EP untuk kelas *Point* ditunjukkan pada **Tabel 4**.

```
1  public class Point {
2      int x, y;
3      ArrayList observers = new ArrayList();
4      public void setX(int x){
5          this.x = x;
6          this.informObservers();
7      }
8      public void setY(int y){
9          this.y = y;
10         this.informObservers();
11     }
12     public void setXY(int x, int y){
13         this.x = x;
14         this.y = y;
15         this.informObservers();
16     }
17     public void attachObservers(Observer observer){
18         observers.add(observer);
19     }
20     public void detachObservers(Observer observer){
21         observers.remove(observer);
22     }
23     public void informObservers(){
24         for (Iterator it = observers.iterator();it.hasNext();)
25             ((Observer)it.next()).update(null, it);
26     }
27 }
```

Gambar 5. Kode kelas *Point*.

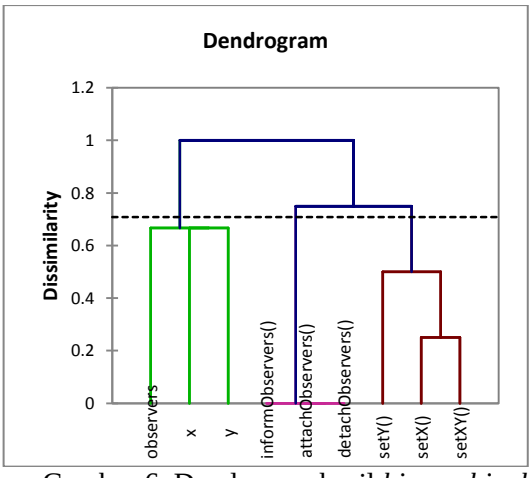
Jurnal Sains, Teknologi dan Industri, Vol. 12, No. 1, Desember 2014, pp. 136 - 145
ISSN 1693-2390 print/ISSN 2407-0939 online

TABEL 4 MATRIKS EP KELAS *POINT*.

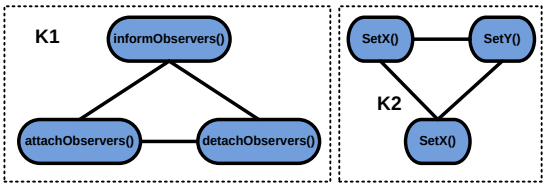
	setX()	setY()	setXY()	attachObservers()	detachObservers()	informObservers()	x	y	observers
setX()	0	0	0	0	0	1	1	0	1
setY()	0	0	0	0	0	1	0	1	1
setXY()	0	0	0	0	0	1	1	1	1
attachObservers()	0	0	0	0	0	0	0	0	1
detachObservers()	0	0	0	0	0	0	0	0	1
informObservers()	0	0	0	0	0	0	0	0	1
X	1	0	1	0	0	0	0	0	0
Y	0	1	1	0	0	0	0	0	0
observers	1	1	1	1	1	1	0	0	0

Hasil *clustering* dengan menggunakan *hierarchical clustering* terlihat pada dendogram di**Gambar 6**. Pada dendogram tersebut terlihat bahwa *method-method* di kelas *Point* dibagi menjadi dua buah *cluster*, K_1 = {*informObservers*(), *attachObservers*(), *detachObservers*()} dan K_2 = {*setX*(), *setY*(), *setXY*()}. Dendogram tersebut juga menunjukkan bahwa *instance* variabel dikelompokkan dalam satu *cluster* khusus.

Gambar 7merupakan graf yang menunjukkan koneksi antar *method* setelah dilakukan proses peng-*cluster*-an. Pada gambar tersebut terlihat hanya ada 6 buah simpul yang terhubung sehingga $|E_A|$ = 6. Karena jumlah*method* 6 buah maka *NP*= 15. Dengan menggunakan rumus yang telah dijelaskan sebelumnya didapat COH_{CL}*= 0.4.Perbandingan hasil pengukuran kohesi dengan metode lain [4] [5] [8] dapat dilihat di **Tabel 5**.



Gambar 6. Dendogram hasil *hierarchical clustering* kelas *Point*.



Gambar 7. Graf kelas *Point*setelah di-*cluster*.

TABEL 5
HASIL PERHITUNGAN KOHESI KELAS *POINT*.

Metriks	LCOM5 [5]	TCC [4]	COH_{CL} [8]	COH_{CL}*
Nilai	0.73	0.33	0.4	0.4

COH_{CL}* merupakan hasil perhitungan dengan pendekatan yang ditawarkan penelitian ini. Pada **Tabel 5** terlihat penggunaan LCOM5 menunjukkan bahwa kelas *point* memiliki kohesi yang baik, yaitu 0.7. Sedangkan dengan menggunakan TCC, COH_{CL} dan COH_{CL}* terlihat kohesi pada kelas *Point* rendah, yaitu masing-masing 0.33, 0.4 dan 0.4.Pada **Gambar 7** terlihat sebenarnya *method-method* di kelas *Point* dibagi menjadi 2 buah bagian, yaitu *method-method* yang memanajemen *point* (*x* dan *y*) dan *method-method* yang memanajemen *instance* variabel *observers*. Sebenarnya teknik pengukuran kohesi dengan *cluster* sebelumnya [8] juga telah berhasil membagi *method-method* menjadi bagian seperti ini, tetapi teknik yang dilakukan penelitian ini lebih rapi mengelompokkan *instance* variabel dalam kelompok khusus seperti yang terlihat pada dendogram di **Gambar 6**.

B. Percobaan pada Kelas TangleStack

Pada percobaan ini akan diukur kohesi pada kelas *TangleStack*. Seperti yang terlihat pada gambar 8, kelas *TangelStack* terdiri dari 5 buah *instance* variabel (*_top*, *_elements*, *s_size*, *_label* dan *_text*) dan 8 buah *method* (*TangleStack*(), *toString*(), *display*(), *push*(), *pop*(), *top*(), *isFull*() dan *isEmpty*()). Karena *method* *TangleStack*() merupakan

*constructor*pada kelas *TangleStack*, maka *method* tersebut diabaikan dalam perhitungan kohesi [4]. Berikut*instance* variabel atau *method* yang digunakan masing-masing *method* di kelas *TangleStack*: *toString*() = {*_top*, *_elements*}, *display*() = {*_text*}, *push*() = {*_top*, *_elements*, *display*(),*isFull()*}, *pop*() = {*_top*, *display*(), *isEmpty()*}, *top*() = {*_top*, *_elements*, *isEmpty()*}, *isFull*() = {*_top*, *s_size*} dan *isEmpty*() = {*_top*}.

*Method**push*() memanggil *metho-display*() dan *isFull*() dalam tubuhnya. Oleh sebab itu semua *instance* variabel yang ada *method display*() dan *isFull*() yang belum ada di *method push*() ditambahkan kedalam*methodpush*(), sehinggaproperti *method push()*tersebut menjadi:*push*() = {*_top*, *_elements*, *display*(), *isFull*(), ***s_size***, ***_text***}.*Methodpop*() memanggil *methoddisplay*() dan *isEmpty*() dalam tubuhnya. Karena *instance* variabel yang digunakan di *method isEmprty*() telah digunakan oleh *method pop*(), maka tidak perubahan properti *methodpop*() karena *method isEmpty*(). Sedangkan pada *method display*() terdapat *instance* variabel yang belum digunakan oleh *method pop*(), sehingga properti *method pop()* tersebut menjadi:*pop*() = {*_top*, *display*(), *isEmpty*(), ***_text***}.*Method top*() menggunakan *methodisEmpty*(), tetapi *instance* variabel yang digunakan oleh *method isEmpty*() telah digunakan *method top*(), oleh sebab itu tidak ada perubahan properti pada *method top*(). Matriks EP untuk kelas *TangleStack* ditunjukkan pada **Tabel 6**.

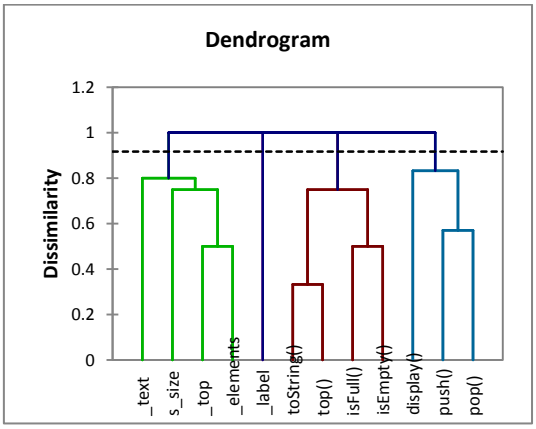
```
public class TangleStack {
    private int _top = -1;
    private Object[] _elements;
    private final int S_SIZE =10;
    private JLabel _label = new JLabel("Stack");
    private JTextField _text = new JTextField(20);
    public TangleStack(JFrame frame){
        _elements = new Object[S_SIZE];
        frame.getContentPane().add(_label);
        _text.setText("[]");
        frame.getContentPane().add(_text); }
    public String toString(){
        StringBuffer result = new StringBuffer("[");
        for (int i = 1; i <= _top; i++){
            result.append(_elements[i].toString());
            if (i != _top) result.append(", ");
        }
        return result.toString(); }
    public void display(){
        _text.setText(toString()); }
    public void push(Object element) throws PreConditionException{
        if (isFull()) throw new PreConditionException("push when stuck full.");
        _elements[++_top] = element;
        display(); }
    public void pop() throws PreConditionException{
        if (isEmpty()) throw new PreConditionException("pop when stuck empty.");
        display(); }
    public Object top() throws PreConditionException{
        if (isEmpty()) throw new PreConditionException("top when stuck empty.");
        return _elements[_top]; }
    public boolean isFull(){
        return (_top == S_SIZE-1); }
    public boolean isEmpty(){
        return (_top == 0); }
}
```

Gambar 8. Kode kelas *TangleStack*.

Hasil peng-*clusteri*-ankelas *TangleStack* dengan menggunakan *hierarchical clustering* terlihat pada dendogram di **Gambar 9**. Pada dendogram tersebut terlihat bahwa *method-method* di kelas *TangleStack* dibagi menjadi dua buah *cluster*, K_1 = {*toString*(), *top*(), *isFull*(), *isEmpty*()} dan K_2 = {*display*(), *push*(), *pop*()}. Dendogram tersebut juga menunjukkan bahwa *instance* variabel dikelompokkan dua buah*cluster*, yaitu kelompok *instance* variabel yang digunakan pada *method-method* yang ada di matriks EP dan kelompok *instance* variabel yang tidak digunakan sama sekali karena hanya digunakan pada *method constructor*. Sebagaimana yang telah dijelaskan sebelumnya, *method constructor* diabaikan dalam perhitungan kohesi.

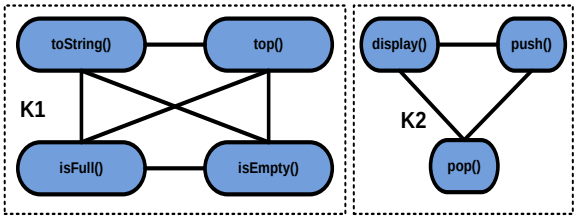
TABEL 6 HASIL PERHITUNGAN KOHESI KELAS *TANGLESTACK*.

	toString()	**display()**	**push()**	**pop()**	**top()**	**isFull()**	**isEmpty()**	**_top**	**_elements**	**s_size**	**_label**	**_text**
toString()	0	0	0	0	0	0	0	1	1	0	0	0
display()	0	0	0	0	0	0	0	0	0	0	0	1
push()	0	1	0	0	0	1	0	1	1	1	0	1
pop()	0	1	0	0	0	0	1	1	0	0	0	1
top()	0	0	0	0	0	0	1	1	1	0	0	0
isFull()	0	0	0	0	0	0	0	1	0	1	0	0
isEmpty()	0	0	0	0	0	0	0	1	0	0	0	0
_top	1	0	1	1	1	1	1	0	0	0	0	0
_elements	1	0	1	0	1	0	0	0	0	0	0	0
s_size	0	0	1	0	0	1	0	0	0	0	0	0
_label	0	0	0	0	0	0	0	0	0	0	0	0
_text	0	1	1	1	0	0	0	0	0	0	0	0



Gambar 9. Dendogram hasil *hierarchical clustering* kelas *TangleStack*.

Gambar 10 merupakan graf yang menunjukkan koneksi antar *method* setelah dilakukan proses peng-*cluster*-an pada kelas *TangleStack*. Pada gambar tersebut terlihat ada 9 buah simpul yang terhubung sehingga $|E_A| = 9$. Karena jumlah *method* 7 buah maka *NP* = 21. Dengan menggunakan rumus yang telah dijelaskan sebelumnya didapat COH_{CL}*= 0.33. Perbandingan hasil pengukuran kohesi dapat dilihat di **Tabel 5**.

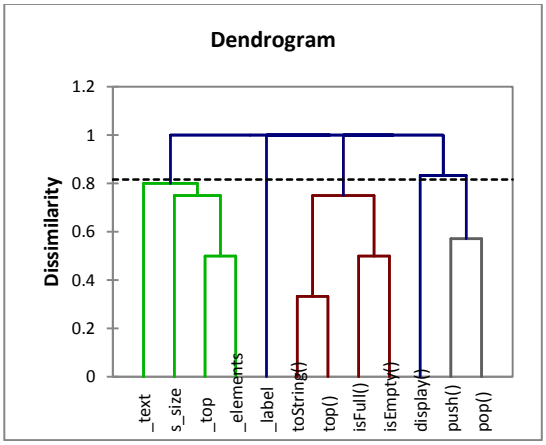


Gambar 10. Graf kelas *TangleStack*setelah di-*cluster*.

TABEL 7
HASIL PERHITUNGAN KOHESI KELAS *TANGLESTACK*.

Metriks	**LCOM5 [5]**	**TCC [4]**	**COH_{CL} [8]**	**COH_{CL} ***
Nilai	0.7	1	0.71	0.33

Pada **Tabel 7** terlihat penggunaan LCOM5, TCC, COH_{CL} menunjukkan bahwa kelas *TangleStack* memiliki kohesi yang baik, yaitu 0.7, 1 dan 0.71. Sedangkan dengan menggunakan COH_{CL}* terlihat kohesi pada kelas *TangleStack*rendah, yaitu hanya 0.33. Ini karena proses *hierarchicalclustering* menghasilkan dua kelompok yang mempunyai jumlah elemen yang hampir sama, sehingga jumlah sisinya semakin sedikit. Tetapi perlu diketahui bahwa proses ini berhasil membagi *method-method* di kelas *TangleStack* berdasarkan fungsi umumnya. Kelompok pertama (*toString*(), *top*(), *isFull*() dan*isEmpty*())adalah kelompok*method* yang hanya memanfaatkankeadaan *stack* tanpa mengubah isi di dalam *stack*.Kelompok kedua (*display*(), *push*() dan *pop*()) adalah kelompok *method* yang bisa mengubah isi *stack*, yaitu *pop*() untuk mengeluarkan satu elemen dari *stuck* dan push() untuk mengisi satu elemen ke dalam *stuck*. Sedangkan*method display*() memang tidak berpengaruh pada isi *stuck* danseharusnya *method* inimemang berada pada *cluster* tersendiri. Seandainya teknik pemotongan untuk menentukan jumlah *cluster*bisa lebih tepat, sehingga bisa menjadi seperti pada dendogram di **Gambar 11**, maka*method display*() akan berada di *cluster* tersendiri, karena memang *method display*() mempunyai fungsi yang berbeda dari *method-method* lainnya, yaitu menampilkan isi *stuck* ke dalam *text field*.Dendogram di **Gambar 11** tersebut merupakan hasil *hierarchical clustering* dengan jumlah *cluster* yang telah ditentukan sebelumnya secara manual, yaitu 5 buah *cluster*.



Gambar 11. Dendogram hasil *hierarchical clustering* kelas *TangleStack*dengan penentuan jumlah *cluster* manual, yaitu 5 buah *cluster*.

KESIMPULAN DAN SARAN KEDEPAN

Studi ini bertujuan untuk mengukur kohesi pada sebuah kelas menggunakan *hierarchical clustering* dengan mempertimbangkan hubungan tidak langsung antara*method* dan *instance* variabel. Pengukuran kohesi dengan*hierarchical clustering*sebenarnya telah dilakukan oleh peneliti sebelumnya [8], tetapi pengukuran kohesi tersebut dilakukan tanpa mempertimbangkan hubungan tidak langsung tersebut. Penelitian ini mengubah matriks EP

pada penelitian sebelumnya [8] yang tidak memasukkan hubungan tidak langsung antara *method* dan *instance* variabel menjadi memasukkan hubungan tersebut sebagai salah satu parameterpengisian elemen matriks EP.

Hasil percobaan menunjukkan bahwa teknik ini tidak hanya dapat digunakan untuk pengukuran kohesi, tetapi teknik ini juga bisa membagi *method-method* disebuah kelas menjadi kelompok-kelompok menurut fungsi umumnya. Teknik ini juga berhasil membagi*instance* variabel menjadi kelompok-kelompok menurut penggunaannya.

Kami menyadari studi ini hanya dilakukan pada kelas yangsederhana dan jumlah kelas yang digunakan pada percobaan juga sedikit. Ini karena keterbatasan waktu kami dalam pengerjaan makalah ini.Oleh sebab itu, kami menyarankan agarteknik ini dicoba pada kelas yang lebih kompleks dengan jumlah kelas yang jauh lebih banyak supaya kemangkusan teknik ini bisa lebih teruji. Kami juga menyarankan mencari teknik penentuan jumlah *cluster* yang lebih tepat untuk teknik ini, sehingga kasus yang terjadi dalam percobaan 2 bisa teratasi

DAFTAR PUSTAKA

S.R. Chidamber dan C.F. Kemerer, "*Towards a Metrics Suite for Object Oriented Design*," Proc. *SixthACMConf. Object Oriented Systems, Languages and Applications*, hal. 197-211, 1991..

S.R. Chidamber dan C.F. Kemerer, "*A Metrics Suite for Object-Oriented Design*," *IEEE Trans. Software Eng*., vol. 20, no. 6, hal. 476-493, June 1994.

M. Hitz dan B. Montazeri, "*Measuring Coupling and Cohesion in Object-Oriented Systems*," *Proc. Symp. Applied Corporate Computing*, 1995.

J.M. Bieman dan B.-K. Kang, "*Cohesion and Reuse in an Object-Oriented System*," *Proc. Symp. Software Reusability*, hal. 259-262, 1995.

C. Henderson-Sellers, Software Metrics. Prentice Hall, 1996.

L.C. Briand, J.W. Daly, dan J.K. Wust, "*A Unified Framework for Cohesion Measurement in Object-Oriented Systems*," *Empirical Software Eng*., vol. 1, no. 1, hal. 65-117, 1998.

H.-S. Chae, Y.-R. Kwon, dan D.-H. Bae, "*A Cohesion Measure for Object Oriented Classes*," *Software Practice and Experience*, vol. 30, no. 12, hal. 1405-1431, 2000.

L. Sadoui, M. Badri dan L. Badri, "*Improving Class Cohesion Measurement: Towards a Novel Approach Using Hierarchical Clustering*," *Journal of Software Engineering and Application*, vol 5, hal. 449-458, **2012**.

S.M. Chandrika, E.S. Babu dan N. Srikanth, "*Conceptual Cohesion of Classes in Object Oriented Systems*," *International Journal of Computer Science and Telecommunications*, vol. 2, hal. 38-44, **2011**.

H.S. Chae dan Y.R. Kwon, "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables,"IEEE Transactions on Software Engineering, vol. 30, no. 11, hal. 826-832. 2004.

M.H. Noor dan M. Hariadi, "*Image Cluster* Berdasarkan Warna untuk Identifikasi Kematangan Buah Tomat dengan Metode *Valley Tracing*," Seminar Nasional Informatika. ISSN: 1979-2328. 2009.

St. Hanenberg and R. Unland, "*A Proposal for Classifying Tangled Code*," 2*ndAOSD-GI Workshop*, Bonn, 21- 22 Februari 2002.

M. J. T. Monteiro, "*Refactorings to Evolve Object-Ori- ented Systems with Aspect-Oriented Concepts*," Ph.D. Thesis, Universidade do Minho, Minho, 2005.