ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

Mengukur Tingkat *Reusability* dan *Efficiency* dari Kode Program dengan Pendekatan *Fuzzy Logic*

Arwin Halim[1], Alex Xandra Albert Sim[2], Gabyola[3], Hartono[4]
Program Studi Teknik Informatika, STMIK Mikroskil
Jl. Thamrin No. 122, 124, 140 Medan 20212
Medan, Indonesia
arwin@mikroskil.ac.id[1], alex.sim@mikroskil.ac.id[2], gabyolacrescent@gmail.com[3], hartono.peng@gmail.com[4]

***Abstrak*—Kode program merupakan salah satu keluaran dalam proses pengembangan sistem. Kualitas dari kode program dapat digunakan untuk merepresentasikan kualitas sistem secara keseluruhan. Salah satu cara untuk mengukur kualitas dari program adalah menghitung nilai metrik kualitas. Nilai-nilai dari metrik kualitas sulit dimengerti karena memiliki makna tersendiri dan bersifat tidak tentu. Pada penelitian ini, faktor kualitas eksternal berupa *reusability* dan *efficiency* diukur melalui pendekatan *fuzzy logic* Mamdani. Masukan dari *fuzzy logic* diperoleh dari hasil perhitungan CK Metrics dari kode program berorientasi objek. Keluaran dari penelitian berupa aplikasi yang mampu menghitung nilai metrik kualitas eksternal dengan menghasilkan nilai kuantitatif dan kualitatif dari kode program, sehingga dapat digunakan untuk membandingkan dua atau lebih kode program.**

Keywords—efficiency; fuzzy logic; kode program; reusability

I. PENDAHULUAN

Berbagai usaha telah dilakukan untuk menyediakan pedoman yang baik dalam mengukur kualitas perangkat lunak menunjukkan pentingnya peran perangkat lunak dalam kehidupan sehari-hari [1]. Kualitas perangkat lunak dapat diukur menggunakan *software quality metrics*. Hasil pengukuran tersebut digunakan untuk menghitung kualitas eksternal perangkat lunak, seperti *maintainability, flexibility, testability, reusability, reliability, efficiency* dan *usability* [2]. Faktor kualitas eksternal menunjukkan aspek kualitas tertentu yang dapat dinilai dari perangkat lunak. Penelitian ini berfokus pada *reusability* dan *efficiency*. *Reusability* menunjukkan kemampuan bagian program yang dapat digunakan kembali. *Efficiency* menunjukkan daya guna program terhadap sumber daya dari kode program untuk menjalankan suatu fungsi [2]. Perangkat lunak berorientasi objek yang berkualitas ditunjukkan dengan nilai *reusability* dan *efficiency* yang tinggi.

Software quality metrics dapat digunakan untuk mengukur kualitas perangkat lunak terstruktur dan berorientasi objek. Kualitas perangkat lunak terstruktur dihitung menggunakan metrik tradisional seperti *Lines of Codes* (LOC), *Cyclomatic Complexity* dan lain-lain. Pada perangkat lunak berorientasi objek, pengukuran terpusat pada bagian-bagian dari *class* seperti data dan prosedur. Menurut Prather dan Weyuker, penilaian kualitas berorientasi objek lebih mudah dinilai secara matematis dibandingkan dengan pemrograman terstruktur [3].

Berbagai metrik untuk pengukuran program berorientasi objek telah diusulkan, seperti Chidamber-Kemerer Metrics (CK Metrics) [3], Lorenz-Kidd Metrics, MOOD, dan QMOOD. Salah satu OO Metrics yang banyak digunakan adalah CK Metrics. Penelitian Mago dan Kaur [4] menunjukkan nilai-nilai dari CK Metrics dapat digunakan untuk mengukur kualitas eksternal perangkat lunak. Penelitian Laird dan Brennan [1] menunjukkan CK Metrics adalah metrik yang berguna jika didefinisikan dengan tepat. Permasalahannya adalah sulitnya mengartikan nilai-nilai dari CK Metrics dengan tepat. Penelitian Stamelos dkk [5] menggunakan pendekatan statistik untuk menentukan kualitas eksternal berdasarkan metrik tradisional pada perangkat lunak berkode bebas seperti *testability*, *simplicity*, *readability* dan *self-descriptiveness.* Penelitian Mago dkk [6] mengusulkan pendekatan *fuzzy* dengan kurva trapesium untuk menentukan nilai tunggal dari kualitas eksternal perangkat lunak berorientasi objek.

Pada penelitian ini, nilai metrik dihitung berdasarkan kode program perangkat lunak berorientasi objek dengan CK Metrics. Program perangkat lunak berorientasi objek dipilih karena pendekatan ini telah berkembang dan digunakan untuk mengembangkan sistem berskala besar sesuai dengan batasan waktu dan biaya [7]. Nilai metrik kualitas digunakan sebagai masukan pada *Fuzzy Inference System* untuk mendapatkan kesimpulan berupa nilai yang menunjukkan faktor kualitas *reusability* dan *efficiency*. Pendekatan *fuzzy* mampu menarik kesimpulan yang lebih bervariatif [8], sehingga memungkinkan pengembang untuk mengukur kualitas eksternal perangkat lunak dengan tepat.

Sistematika penulisan dimulai dari pendahuluan. Bagian kedua menjelaskan landasan teori yang digunakan dalam penelitian dan dilanjutkan dengan penjabaran metode

ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

penelitian. Bagian keempat menunjukkan hasil dan pengujian dari penelitian. Pada bagian terakhir berisi kesimpulan.

II. Tinjauan Pustaka

A. Chidamber-Kemerer Metrics

Chidamber-Kemerer Metrics (CK Metrics) [3] merupakan salah satu *Object Oriented Metric* yang dapat digunakan untuk mengukur kualitas desain sebuah program. CK Metrics diusulkan oleh Shyam R. Chidamber dan Chris F. Kemerer pada tahun 1994. CK Metrics terdiri dari 6 metrik yang dijadikan parameter dalam mengukur kualitas program. Keenam metrik tersebut adalah:

1. *Weight Method per Class* (WMC).

   WMC merupakan metrik yang berfokus pada kompleksitas dari *method* pada sebuah *class*. Kompleksitas dapat dihitung dengan rumus *cyclomatic complexity*. WMC menjumlahkan semua nilai kompleksitas dalam semua *method* di dalam sebuah *class* seperti Persamaan 1.

$$WMC = \sum_{i=1}^{n} c_i \quad (1)$$

   dimana:
   n = jumlah *method* dalam sebuah *class*
   ci = *cyclomatic complexity* dari sebuah *method i*

2. *Depth of Inheritance Tree* (DIT)

   DIT merupakan jarak *node* ke *root* dalam *class* yang memiliki *inheritance* atau dengan kata lain berapa banyak *inheritance* yang digunakan dalam *class* tersebut. Semakin tinggi nilai DIT menunjukan desain yang lebih kompleks dan *reusability* yang lebih tinggi, tetapi nilai DIT yang tinggi juga menunjukkan tingkat *efficiency* yang rendah.

3. *Number of Children* (NOC)

   NOC merupakan jumlah sub*class* langsung yang terdapat dalam sebuah *class*. NOC mengukur berapa banyak sub*class* yang meng-*inherit method* dari *parent class*. Nilai NOC yang tinggi tidak hanya menunjukkan *reusability* yang tinggi, tetapi juga kemungkinan adanya sub*class*ing yang tidak tepat yang mengurangi nilai *efficiency*.

4. *Coupling Between Object Classes* (CBO)

   CBO menghitung *coupling* non-*inheritance* antara suatu *class* terhadap *class* lain, contohnya suatu *class* menggunakan *method* atau variabel dari *class* lain. Jika suatu *class* memanggil beberapa *method* maupun variabel dari sebuah *class* lain, maka jumlah relasi tetap dihitung

   1. Nilai CBO yang tinggi menunjukkan kurangnya *reusability*, *efficiency*, dan *maintainability.*

5. *Response For a Class* (RFC)

   RFC adalah himpunan *method* yang dapat dijalankan sebagai respon dari objek dalam sebuah *class*. RFC dihitung menggunakan Persamaan 2.

$$RFC = \{M\} \cup \{R_i\} \quad (2)$$

   dimana:
   M = himpunan *method* yang dipanggil dalam *class*
   R = himpunan *method* yang dipanggil oleh *method* i

6. *Lack of Cohesion of Methods* (LCOM)

   LCOM menghitung selisih antara jumlah *method* dalam sebuah *class* yang memiliki parameter atau *return type*. Semakin besar nilai LCOM, semakin besar pula kompleksitas *class* tersebut dan berpengaruh pada rendahnya tingkat *reusability* dan *efficiency*. LCOM dapat dihitung sesuai Persamaan 3 dan 4.

$$\sigma() = \{I_1\} \cap \{I_2\} \quad (3)$$

   dimana:
   I1= himpunan parameter atau *return type* yang dimiliki *method* 1
   I2= himpunan parameter atau *return type* yang dimiliki *method* 2
   Catatan: σ() dilakukan terhadap semua perpaduan *method* dalam satu *class*.

$$LCOM = \begin{cases} 0, & P < Q \\ P - Q, & P \geq Q \end{cases} \quad (4)$$

   dimana:
   P = jumlah σ( ) dengan nilai 0
   Q = jumlah σ( ) dengan nilai lebih dari 0

CK Metrics dapat menilai kompleksitas program, besar *class*, penggunaan *inheritance*, perpaduan dalam *class*, dan hubungan dengan *class* lainnya. Sedangkan hubungan metrik-metrik tersebut dengan faktor kualitas eksternal perangkat lunak dapat dilihat dalam Tabel 1 [4].

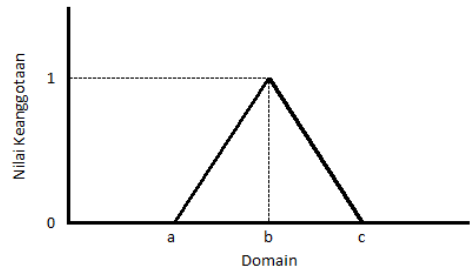
ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

Tabel 1. Hubungan CK Metrics dan Faktor Kualitas Eksternal

CK Metrics	Kualitas Eksternal	Konsep OOP
WMC	*Maintainability, Understandability, Reusability*	*Class/Method*
RFC	*Understandability, Testability, Maintainability*	*Class/Method*
LCOM	*Reusability, Efficiency*	*Class/Method*
CBO	*Reusability, Efficiency*	*Coupling*
DIT	*Reusability, Efficiency, Testability*	*Inheritance*
NOC	*Reusability, Efficiency, Testability*	*Inheritance*

B. *Fuzzy Logic*

Sistem pengambilan keputusan konvensional menyimpulkan suatu nilai dengan memasukannya ke dalam suatu himpunan tegas (*crisp sets*) yang memiliki fungsi keanggotaan dengan dua kemungkinan nilai keanggotaan, yaitu 1 yang berarti nilai tersebut merupakan anggota dari himpunan itu, dan 0 yang berarti nilai tersebut bukan anggota dari himpunan itu. Sedangkan dalam himpunan *fuzzy*, fungsi keanggotaannya bisa memiliki nilai dengan range 0 sampai 1. Perhitungan nilai keanggotaan dalam himpunan *fuzzy* dapat direpresentasikan dalam beberapa cara, seperti representasi linear, kurva trapesium, kurva segitiga, dan lain-lain. Gambar 1 merepresentasikan nilai keanggotaan dengan kurva segitiga pada *fuzzy logic*.



Gambar 1. Nilai Keanggotaan dengan Kurva Segitiga

Perhitungan nilai keanggotaan untuk kurva segitiga dapat dilihat di Persamaan 5.

$$f(x) = \begin{cases} 0, & x \le a \text{ atau } x \ge c \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{d-x}{d-c}, & x \ge c \end{cases} \quad (5)$$

Beberapa himpunan *fuzzy*, dapat digabung menggunakan operator-operator dasar seperti operator AND dan operator OR. Sistem *fuzzy* menggunakan fungsi implikasi dan aturan *fuzzy* sebagai dasar dalam pengambilan keputusan. Bentuk umum dari aturan yang digunakan adalah IF-THEN *rule*. Untuk mendapat nilai *output*, fungsi implikasi digunakan untuk memotong kurva fungsi keanggotaan *output* berdasarkan aturan-aturan *fuzzy* yang ada. Salah satu cara fungsi implikasi yang dapat digunakan adalah metode *Min (minimum)*. Metode ini memotong kurva fungsi keanggotaan *output* berdasarkan himpunan *input* yang memiliki nilai keanggotaan terkecil (operator AND).

Metode Mamdani [7] diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Metode ini sering juga disebut metode *Max-Min*, karena menggunakan komposisi aturan *Max* (operator OR) dan fungsi implikasi *Min* (operator AND). Tahapan-tahapan dalam metode ini antara lain:

a. Pembentukan himpunan *fuzzy*
Pada metode ini, variabel *input* dan *output* dinyatakan ke dalam himpunan *fuzzy*.

b. Aplikasi fungsi implikasi
Fungsi implikasi yang digunakan dalam metode ini adalah *Min*.

c. Komposisi aturan
Komposisi aturan adalah penggabungan dari kurva-kurva yang didapat dari hasil perhitungan fungsi implikasi. Pada metode ini yang digunakan adalah metode *Max* (*maximum*), oleh karena itu metode Mamdani juga disebut dengan metode *Max-Min*. Metode *Max* mengambil nilai keanggotaan tertinggi dari setiap kurva hasil perhitungan fungsi implikasi kemudian mengaplikasikannya menjadi sebuah fungsi keanggotaan dengan menggunakan operator OR, atau dapat ditulis dalam Persamaan 6.

$$\mu_{sf}[X_i] \leftarrow \max(\mu_{sf}[X_i], \mu_{kf}[X_i]) \quad (6)$$

dimana:
$\mu_{sf}[xi]$ = nilai keanggotaan *input* sampai aturan ke-*i*
$\mu_{kf}[xi]$ = nilai keanggotaan *output* aturan ke-*i*

d. *Defuzzification*
Defuzzification adalah proses untuk mendapatkan sebuah nilai tunggal dari nilai-nilai keanggotaan yang diperoleh dari komposisi aturan *fuzzy*. Ada beberapa metode yang dapat digunakan dalam *defuzzification*, antara lain metode *centroid*, *weighted average*, bisektor, *mean of maximum*, *largest of maximum*, dan *smallest of maximum*. Metode yang digunakan dalam penelitian ini adalah metode *weighted average* yang dihitung menggunakan Persamaan 7.

$$Y = \frac{\sum CCi \times Ti}{\sum Ti} \quad (7)$$

dimana:

ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

Y = nilai tunggal hasil *defuzzification*
CCi = nilai tengah kurva output-i
Ti = nilai keanggotaan kurva output-i

III. METODOLOGI PENELITIAN

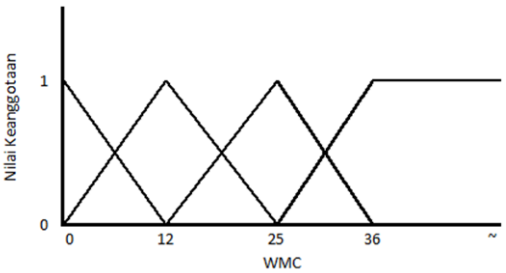
Tahapan pengukuran faktor kualitas eksternal berupa *reusability* dan *efficiency* adalah:

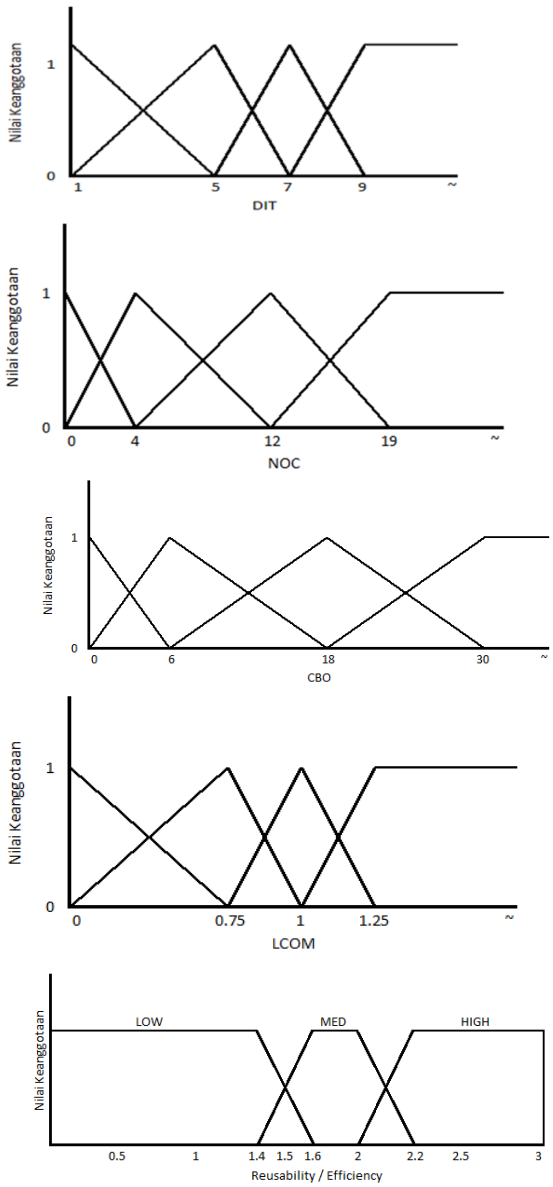
1. Pengumpulan data.
   Pada tahapan ini dikumpulkan proyek-proyek C# berkode bebasyang memiliki versi yang berbeda dari berbagai sumber. Proyek-proyek inilah yang akan diukur nilai kualitasnya.

2. Analisis Proses
   Pada tahapan ini, dilakukan perhitungan nilai CK Metrics untuk setiap *class* pada proyek perangkat lunak, kemudian hasil pengukuran digunakan sebagai masukan pada *Fuzzy Inference System* untuk mengetahui nilai kualitas eksternal proyek. Faktor kualitas eksternal *reusability* dipengaruhi oleh metrik WMC, LCOM, CBO, DIT dan NOC. Faktor kultias eksternal *efficiency* dipengaruhi oleh metrik LCOM, CBO, DIT dan NOC. Tahapan dalam metode Mamdani *Fuzzy Inference System* yang diadopsi adalah:

   a. Pembentukan himpunan dan aturan *fuzzy*. Dalam tahap ini dibentuk himpunan *fuzzy* dan kurva keanggotaan dari tiap-tiap *input* dan *output*. Setiap *input* memiliki empat nilai yaitu *very low*, *low*, *medium*, dan *high*. Sedangkan setiap *output* memiliki tiga nilai yaitu *low*, *medium*, dan *high* [4]. Selain itu juga dibentuk himpunan *fuzzy* berdasarkan pengaruh metrik-metrik CK Metrics terhadap *reusability*, dan *efficiency*.

   b. Aplikasi fungsi implikasi. Pada tahap ini dicari nilai keanggotaan setiap input mengunakan fungsi kurva segitiga. Gambar 2 dan Gambar 3 menunjukkan contoh kurva keanggotaan untuk metrik kualitas dan faktor kualitas eksternal.





Gambar 2. Nilai Keanggotaan untuk Metrik WMC, DIT, NOC, CBO, LCOM, *Reusability* dan *Efficiency*

   c. Komposisi Aturan. Komposisi aturan memotong kurva *output* sesuai dengan penggabungan hasil yang didapat dari tahap sebelumnya menggunakan metode max
   d. *Defuzzification*. Proses mendapat sebuah nilai tunggal dari hasil yang sudah ada dengan menggunakan metode *weighted average.*

3. Pengembangan Aplikasi
   Pada tahap ini dilakukan pengembangan aplikasi pengukuran kualitas eksternal *reusability* dan *efficiency* dengan menggunakan bahasa pemrograman C#.

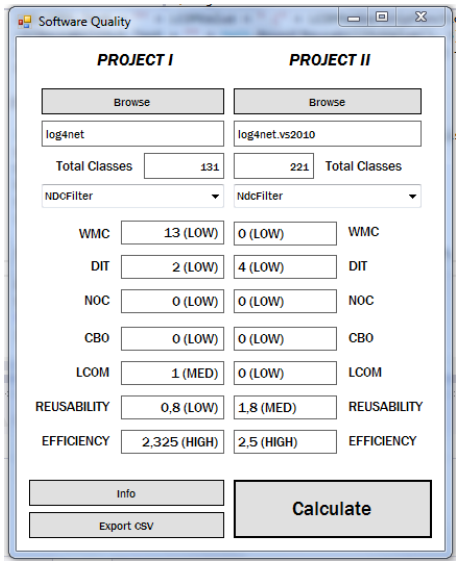
ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

4. Pengujian
   Pada tahap ini dilakukan pengujian aplikasi yang telah dikembangkan dengan aplikasi berkode sumber bebas yang telah dikumpulkan.

IV. HASIL DAN PENGUJIAN

A. Hasil

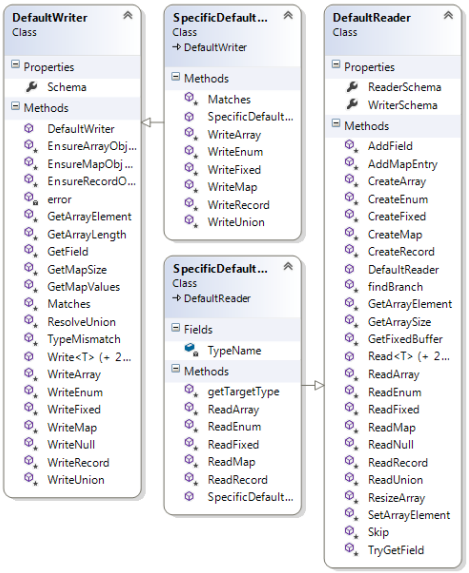
Tampilan utama dari aplikasi yang dikembangkan menerima masukan berupa *solution* dari proyek dengan bahasa pemrograman C#. Gambar 3 menunjukkan contoh keluaran dari perhitungan nilai CK Metrics dan metrik kualitas eksternal *reusability* dan *efficiency*.



Gambar 3. Tampilan Utama Aplikasi Pengukuran Kualitas *Reusability* dan *Efficiency* dengan *Fuzzy Logic*

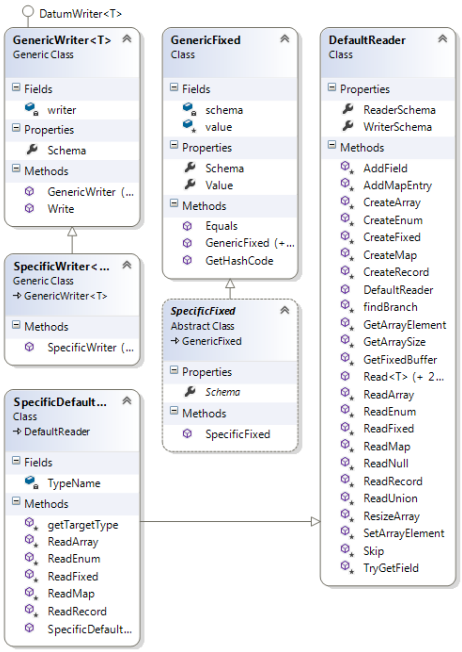
B. Pengujian

Data pengujian yang digunakan adalah Apache Avro C# API. Perangkat lunak Avro merupakan salah satu proyek Apache untuk menserialisasi data. Serialisasi data adalah proses mengubah *object* menjadi bentuk biner atau persamaan linear agar dapat disimpan atau dikirim ke lokasi lain pada jaringan. Versi Avro yang diuji adalah versi 1.6.0 dan versi 1.7.6 yang dapat diunduh dari http://archive.apache.org/dist/avro/. Pada pengujian ini, *class* dengan nama yang sama pada kedua proyek yang dibandingkan terlebih dahulu untuk melihat kualitas *class* yang sama pada versi proyek yang berbeda, kemudian dibandingkan juga secara keseluruhan *class* pada proyek untuk kedua versi yang berbeda tersebut untuk melihat kualitasnya secara keseluruhan. Jadi, total *class* yang sama untuk Avro 1.6.0 dan 1.7.6 adalah 42 *class*. Total *class* Avro 1.6.0 adalah 42 *class* dan Avro 1.7.6 adalah 78 *class*. Gambar 4 menunjukkan sebagian *class diagram* dari Avro versi 1.6.0.



Gambar 4. Sebagian *Class* Diagram dari Avro C# API versi 1.6.0.

Gambar 5 menunjukkan sebagian *class diagram* dari Avro versi 1.7.6.



Gambar 5. Sebagian *Class* Diagram dari Avro C# API versi 1.7.6.

Rangkuman hasil perhitungan metrik kualitas eksternal dari Avro C# API versi 1.6.0 dan 1.7.6 untuk *class* yang sama

ISSN 2087-2658
Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)
Volume 4, Nomor 3, Desember 2015

dapat dilihat pada Tabel 2. Berdasarkan hasil yang diperoleh, kita dapat melihat nilai mininum dan maksimum yang sama untuk *reusability* dan *efficiency* pada kedua proyek. Namun, total *reusability* dan *efficiency* dari Avro 1.6.0 sedikit lebih tinggi dibandingkan dengan Avro 1.7.6. Hal ini berarti terjadinya penurunan kualitas *reusability* dan *efficiency* pada Avro 1.7.6 pada *class* yang sama. Namun, jika kita melihat secara keseluruhan *class* pada Avro 1.7.6 sebanyak 78 *class*, maka rata-rata nilai *reusability* yang diperoleh adalah 1.21067 dan rata-rata nilai *efficiency* adalah 2.23576. Nilai *reusability* Avro 1.7.6 secara keseluruhan lebih baik dari Avro 1.6.0 dengan selisih sebesar 0.02736. Hal tersebut juga terjadi pada *efficiency* dengan selisih sebesar 0.04732. Jadi, secara keseluruhan Avro 1.7.6 mengalami peningkatan kualitas *reusability* dan *efficiency* dibandingkan dengan versi Avro 1.6.0.

Tabel 2. Hasil Pengukuran kualitas *class* yang sama pada Avro C# API

Faktor Kualitas Eksternal		**Versi Perangkat Lunak**	
		Avro 1.6.0	**Avro 1.7.6**
Reusability	Min	0.8	0.8
	Max	1.8	1.8
	Sum	49.69901	47.99316
	Average	1.18331	1.14269
Efficiency	Min	1.4	1.4
	Max	2.5	2.5
	Sum	91.91439	91.73939
	Average	2.18844	2.18271

V. KESIMPULAN

Penelitian ini menghasilkan aplikasi pengukuran kualitas *reusability* dan *efficiency* dengan menggunakan nilai CK Metrics dan *fuzzy inference system Mam*dani. Aplikasi ini dapat memberikan informasi yang jelas mengenai kualitas dari *class* pada perangkat lunak berorientasi objek. Hasilnya berupa nilai kuantitatif dan kualitatif, sehingga dapat dengan mudah digunakan untuk melakukan perbandingan kualitas antara dua kode program yang memiliki fungsi yang sama dan memberikan panduan untuk pengembangan perangkat lunak versi berikutnya.

Referensi

[1] Laird, L. M. dan Brennan, M. C., 2006, Software Measurement and Estimation: A Practical Approach, edisi 1, John Wiley & Sons, Inc.

[2] Cavano, J. P., dan McCall, J. A., 1978, A Framework for the Measurement of Software Quality, Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues, 133-139

[3] Chidamber, S. R., dan Kemerer, C. F., 1994, A Metric Suite for Object Oriented Design, IEEE Transactions on Software Engineering, vol 20, 476-493

[4] Mago, J. dan Kaur, P., 2012, Analysis of Quality of the Design of the Object Oriented Software using *Fuzzy Logic*, International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) , vol 3, 21-25

[5] Stamelos, I., et al, 2002, Code Quality Analysis in Open Source Software Development, Information Systems Journal, vol 12, 43-60

[6] Mago, J., Kaur,S., dan Saurabh, K., 2012, *Fuzzy* Model to Analyze and Interpret Object Oriented Software Design, International Journal of Electrical, Electronics, and Computer, vol 1, 41-46.

[7] O'Docherty, M., 2005, Object-oriented Analysis & Design – Understanding System Development with UML 2.0, West Sussex, England: John Willey & Sons.

[8] Kusumadewi, S., dan Purnomo, H., 2010, Aplikasi Logika *Fuzzy* untuk Pendukung Keputusan Ed. 2, Graha Ilmu.