# PEMERINGKATAN *SOFTWARE* APLIKASI BERDASARKAN PROPERTI KUALITAS DISAIN DAN *METRICS FOR OBJECT ORIENTED SOFTWARE* MENGGUNAKAN *ANALYTIC HIERARCHY PROCESS*

**Efano Hermawan[1] dan Petrus Mursanto[2]**

[1]Magister Teknologi Informasi, Universitas Indonesia, Jl. Salemba Raya 4, Jakarta, 12000, Indonesia
[2]Enterprise Computing Lab - Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI Depok, Jawa Barat, 16424, Indonesia

E-mail: efano.hermawan@gmail.com

**Abstrak**

Sebuah metode untuk mengukur kualitas desain berdasarkan hasil implementasinya dalam Java *source codes* diusulkan dalam penelitian ini. Metode yang diusulkan menggabungkan *Metrics for Object-Oriented Software Engineering* (MOOSE), properti kualitas desain *software* dan konsep *Analytic Hierarchy Process* (AHP). Sebagai studi kasus, metode ini diterapkan pada sejumlah aplikasi ERP yang bersifat *open source* yaitu Adempiere, OpenBravo, Plazma, FreedomERP, dan JAllInOne. Pengukuran MOOSE dilakukan dengan bantuan *tool* CKJM 1.8. Hasil ukur MOOSE dikelompokkan dalam properti kualitas yaitu *efficiency, understandability, reusability, testability* dan *maintainability*. Kombinasi MOOSE dan AHP yang dihasilkan dapat menjadi alat bantu dalam menentukan peringkat kualitas *software* dari aspek orientasi objek.

**Kata Kunci:** *analytic hierarchy process, MOOSE*, *object oriented*, *quality property*

**Abstract**

A method for measuring the quality of the design is based on the results of its implementation in the Java source codes proposed in this study. The proposed method combines Metrics for Object-Oriented Software Engineering (MOOSE), property and the concept of software design quality Analytic Hierarchy Process (AHP). As a case study, this method is applied to a number of applications that are open source ERP is Adempiere, Openbravo, Plazma, FreedomERP, and JAllInOne. MOOSE measurements done with the aid tool CKJM 1.8. MOOSE measuring results are grouped in quality properties that are efficiency, understandability, reusability, testability and maintainability. MOOSE and AHP combination that have been produced can be a useful tool in determining the quality ratings of aspects of object oriented software.

**Keywords:** *analytic hierarchy process, MOOSE*, *object oriented*, *quality property*

## 1. Pendahuluan

Pengembangan piranti lunak dengan menggunakan konsep orientasi objek telah memudahkan *designer* dalam membangun *software*. Konsep orientasi objek juga telah memacu *designer* dalam pengembangan *software* menggunakan teknik dan pola-pola implementasi yang berbeda. Beragamnya pola implementasi orientasi objek juga menimbulkan perbedaan pendapat dalam melihat kualitas sebuah *software*. Namun, berbagai teknik dan pola implementasi tersebut memiliki sejumlah karakteristik baku yang berlaku umum sesuai dengan kaidah perancangan berorientasi objek. Karakteristik baku ini seharusnya dapat dikuantifikasi sehingga menghasilkan seperangkat parameter sebagai alat ukur kualitas sebuah desain berorientasi objek. Pemeringkatan kualitas *software* berdasarkan aspek desain objek pada *software* merupakan bidang penelitian yang masih berkembang hingga saat ini.

Beberapa riset terdahulu telah menghasilkan alat ukur kualitas *software* yang diwujudkan dalam beberapa parameter-parameter seperti MOOD dan MOOD2 – *Metrics for Object-*

*Oriented Design* [1][2], MOOSE – *Metrics for Object-Oriented Software Engineering* [3]. Alat-alat ukur kualitas *software* ini biasa disebut dengan singkat yaitu *Object Oriented* (OO) *Metrics*.

Dalam penerapan OO *Metrics*, kualitas *software* diinterpretasikan dalam parameter-parameter yang sesuai dengan properti OO yang diimplementasikan. Namun dengan nilai-nilai dari parameter yang beragam tersebut, masih sangat sulit untuk menentukan *software* mana yang memiliki kualitas lebih baik. Sehingga diperlukan sebuah metode umum untuk mengkombinasikan keseluruhan nilai tersebut menjadi sebuah nilai yang menginterpretasikan kualitas *software* relatif terhadap *software* yang lain. Dikarenakan domain aplikasi turut mempengaruhi struktur desain dan implementasi, maka perbandingan kualitas seharusnya dilakukan terhadap sejumlah aplikasi dalam domain yang sama. Dengan metode yang diusulkan diharapkan penilaian objektif kualitas *software* dapat dengan mudah dilakukan. Metode ini seharusnya dapat dipergunakan secara umum pada semua jenis *software* seperti aplikasi *desktop*, aplikasi *web*, *class library*, ataupun API.

Penerapan MOOSE dan AHP yang diusulkan telah diujicobakan pada *sample software* ERP *open source* berbasis Java. Alasannya, ERP merupakan sebuah *software* berskala *enterprise* dengan banyak modul yang terintegrasi sehingga sesuai dengan karakteristik orientasi objek yang ingin dicapai oleh penelitian ini. Saat ini banyak ERP yang bersifat *open source* dengan basis bahasa pemrograman yang berbeda-beda seperti PHP, Phyton, Java, dan lain-lain. Dalam penelitian ini peneliti menggunakan ERP berbasis Java karena tingkat kuantitas ERP berbasis Java lebih banyak dibandingkan ERP yang dikembangkan dengan bahasa pemrograman yang lain.

Pengukuran kualitas desain *software* menggunakan parameter MOOSE CK dan *tool* CKJM 1.8. Saat ini telah banyak jenis OO *Metrics*, misalnya yang dibahas dalam [1-3]. MOOSE CK sengaja dipilih karena penelitian [4-6] telah menghasilkan pemetaan antara MOOSE CK dan properti kualitas. CKJM 1.8 dilaporkan oleh [7] memiliki respon yang cukup baik dalam menghitung parameter MOOSE CK.

MOOSE CK merupakan salah satu kumpulan *metric* yang dipergunakan untuk mengukur kualitas sebuah *software* berdasarkan enam parameter dengan melihat pada perspektif *Object Oriented Design* [8]. Enam parameter tersebut adalah *Weighted Methods per Class* (WMC), *Depth of Inheritence Tree* (DIT), *Number of Children* (NOC), *Coupling Between Object Classes* (CBO), *Response for a Class* (RFC), dan *Lack of Cohession Method* (LCOM).

Secara umum karakteristik tiap-tiap parameter MOOSE CK memiliki kecenderungan berbanding terbalik dengan kualitas desain *software*, semakin kecil nilai parameter MOOSE CK maka semakin baik kualitas desain *software*. Selain itu, MOOSE CK berpengaruh pada konsep orientasi objek dan bukan prosedural. Sehingga konsep *inheritance*, *encapsulation* dan *polymorphism* benar-benar mempengaruhi nilai dari tiap-tiap parameter MOOSE CK.

Beberapa penelitan telah menghasilkan model kualitas *software* dan perspektif yang berbeda-beda. McCall dkk. melihat dari tiga perspektif yaitu *product revision*, *product transition*, dan *product operations* [9]. Dari masing-masing perspektif dikembangkan menjadi beberapa faktor kualitas *software* yaitu *maintainability, flexibility, testability, portability, reusability, interoperability, correctness, reliability, efficiency, integrity, usability*. Sehingga total memiliki sebelas faktor kualitas *software*. Boehm dkk. mengusulkan tujuh faktor kualitas *portability, reliability, efficiency, usability, testability, understandability,* dan *flexibility* [10]. ISO juga mengeluarkan model kualitas *software* yang dituangkan dalam ISO1926-1 dan memiliki enam karakteristik kualitas *software functionality, reliability, usability, efficiency, maintainability, portability* yang masing-masing dipecah dalam beberapa subkarakteristik. Masing-masing parameter dalam *metric* memiliki kontribusi yang berbeda pada properti kualitas *software*. Tabel I memperlihatkan hasil pemetaan *metrics* dan properti kualitas menurut [4].

TABEL I
HUBUNGAN PROPERTI KUALITAS *SOFTWARE* DAN *PARAMETER METRIC*

| Properti kualitas | *Metrics* |
|---|---|
| Efficiency | LCOM, CBO,DIT,NOC |
| Complexity | CC (Traditional Metrics) |
| Understandability | WMC, RFC,DIT |
| Reusability | WMC, LCOM, CBO, DIT, NOC |
| Maintainability/ Testability | WMC, RFC, DIT, NOC |

AHP adalah sebuah metode perbandingan berpasangan terhadap beberapa objek yang akan dievaluasi. AHP pertama kali dipublikasikan oleh T.L. Saaty dalam bukunya yang berjudul *"The Analytic Hierarchy Process"* tahun 1980 [11]. Dalam memberikan nilai pada masing-masing objek atau faktor yang akan dievaluasi, AHP memberikan skala 1-9. Semakin besar nilai yang diberikan berarti objek atau faktor tersebut semakin penting. Tedapat mekanisme untuk memastikan konsistensi penilaian perbandingan preferensi antar pasangan parameter yang dievaluasi.

## 2. Metodologi

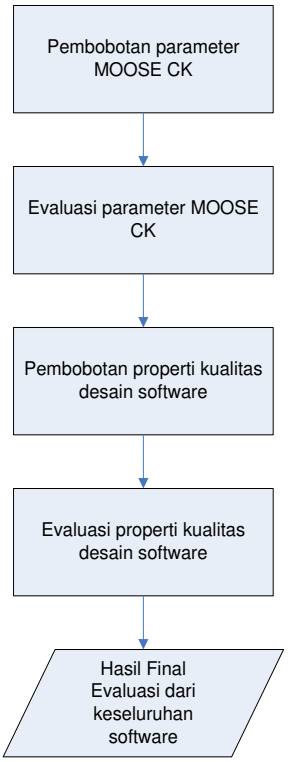
Kebutuhan akan adanya sebuah metode yang dapat mengukur kualitas sebuah *software* menjadi alasan utama dilakukannya penelitian ini. Hasil penelitian diharapkan menghasilkan sebuah metode yang dapat membantu para pengguna menentukan *software* mana yang terbaik dari beberapa *software* yang dievaluasi.

Pada studi kasus ini peneliti menggunakan lima *software* ERP *open source* yaitu Adempiere 3.2, Openbravo 2.5.3, Plazma 0.1.5, FreedomERP 1.1.3.5, dan JAllInOne0.9.0.6. Kelima *software* tersebut dipilih karena memiliki aktifitas yang tinggi serta banyak diminati oleh banyak orang menurut versi www.sourceforge.net. Pengukuran parameter MOOSE CK dilakukan dengan menggunakan aplikasi CKJM 1.8 (Chidamber Kemerer Java Metric).

Untuk mempermudah penjelasan mengenai metode pengukuran kualitas desain, peneliti membagi metode tersebut dalam empat tahap utama, antara lain: pembobotan MOOSE CK, evaluasi MOOSE CK, pembobotan properti kualitas desain, dan evaluasi properti kualitas desain. Tahapan riset dijelaskan pada gambar 1. Hasil akhir dari yang diharapkan dari metode ini adalah nilai kuantitatif masing-masing *software* yang menginterpretasikan kualitas desain aplikasi tersebut. Nilai tersebut merupakan representasi peringkat kualitas aplikasi relatif terhadap yang lainnya.

Pembobotan Parameter MOOSE CK: Pada tahap pembobotan, parameter MOOSE CK dibagi dalam sembilan tingkat. Tujuannya adalah untuk mendapatkan bobot dari masing-masing parameter MOOSE CK. Hasil yang diperoleh dari pembobotan parameter-parameter MOOSE CK harus dicek konsistensinya, dengan tujuan untuk mengukur konsistensi yang diberikan berdasarkan skala preferensi 1 sampai dengan 9.

Perbandingan Nilai Skala Saathy pada Parameter MOOSE CK: Perbandingan parameter-parameter MOOSE CK berpedoman pada skala Saathy, seperti yang di tunjukan pada tabel II. Dalam membandingkan parameter-parameter MOOSE CK harus mempertimbangkan objektifitas dari pengukuran *software*. Pemetaan *metric* dengan *Object Oriented Design Element* seperti dalam tabel III merupakan salah satu *tool* yang dapat digunakan dalam membandingkan parameter-parameter MOOSE CK. Dengan mengetahui orientasi dari masing-masing parameter MOOSE CK, maka pemberian nilai skala dapat disesuaikan dengan objektifitas pengukuran kualitas desain *software*.




Gambar 1. Tahapan utama metode penelitian.

TABEL II
HASIL PERBANDINGAN ANTARA PARAMETER

| | WMC | DIT | NOC | CBO | RFC | LCOM |
|---|---|---|---|---|---|---|
| WMC | 1 | 3 | 4 | 1/4 | 1/7 | 2 |
| DIT | 1/3 | 1 | 3 | 1/5 | 1/8 | 1/2 |
| NOC | 1/4 | 1/3 | 1 | 1/6 | 1/9 | 1/3 |
| CBO | 4 | 5 | 6 | 1 | 1/3 | 3 |
| RFC | 7 | 8 | 9 | 3 | 1 | 4 |
| LCOM | 1/2 | 2 | 4 | 1/3 | 1/4 | 1 |

TABEL III
PEMETAAN *METRIC* DENGAN *OBJECT ORIENTED DESIGN ELEMENT* [12]

| Metric | Object definition | Object attributes | Object communictn |
|---|---|---|---|
| WMC | √ | √ | |
| DIT | √ | | |
| NOC | √ | | |
| RFC | | √ | √ |
| CBO | | | √ |
| LCOM | | √ | |

Dalam studi kasus diberikan contoh dalam pemberian nilai perbandingan dari pasangan masing-masing parameter MOOSE CK. Dalam contoh ini peneliti ingin mengukur *dependency* objek-objek dalam *software*, sehingga parameter yang memiliki orientasi *object communication* memiliki nilai perbandingan lebih besar dibandingkan orientasi yang lain. Tahapan selanjutnya adalah *object attribute*. Elemen ini mewakili konsep *encapsulation* yang dapat mempengaruhi faktor *dependency* objek dalam *software*. Kemudian dilajutkan dengan *object definition*. Hasil perbandingan pada tabel tidak selalu sama tergantung pada objektifitas *software* yang akan diukur.

Dengan preferensi pasangan parameter seperti dalam tabel II, diperoleh nilai bobot masing-masing *metric* seperti disajikan dalam tabel IV.

TABEL IV
BOBOT PARAMETER MOOSE CK

| Metric | Bobot |
|---|---|
| WMC | 0.1149 |
| DIT | 0.0565 |
| NOC | 0.0312 |
| CBO | 0.2396 |
| RFC | 0.4617 |
| LCOM | 0.0963 |

Menghitung *Weighted Sum Vector*: Setelah dihasilkan bobot pada masing-masing parameter MOOSE CK, perlu dipastikan konsistensi preferensi yang telah ditentukan. Cek konsistensi dilakukan dengan menghitung *Weighted Sum Vector* yang kemudian dibagi oleh masing-masing bobot untuk menghasilkan *consistency index*. Dalam contoh ini rata-rata bobot adalah 6.3582. Untuk jumlah parameter enam diperoleh *consistency index* (CI):

$$CI = \frac{\lambda - n}{n-1} \quad (1)$$

n adalah jumlah *item* yang dibandingkan dalam kasus ini n = 6

$$CI = \frac{\lambda - n}{n-1} = \frac{6.3582 - 1}{6-1} = 0.0716 \quad (2)$$

dan selanjutnya diperoleh *consistency ratio* (CR)

$$CR = \frac{CI}{RI} = \frac{0.0716}{1.24} = 0.057 \quad (3)$$

Sesuai dengan konsep AHP, jika *consistency ratio* adalah 0.1 atau kurang, maka perbandingan diatas adalah konsisten; namun apabila *consistency ratio* lebih besar daripada 0.1 maka perbandingan antar parameter harus diulang kembali. Dikarenakan nilai CR 0.057 kurang dari 0.1 maka pembobotan parameter yang dihasilkan pada tabel IV dapat diterima.

Evaluasi Parameter MOOSE CK: Setelah diperoleh parameter MOOSE CK, selanjutnya dilakukan perbandingan antar *software* dengan menggunakan salah satu parameter MOOSE CK. Namun karena nilai parameter MOOSE berbanding terbalik dengan properti kualitas desain *software*, maka perbandingan tersebut perlu dilakukan inversi. Sehingga apabila *software* I WMC = a, *software* II WMC = b, *software* III WMC = c maka diperoleh nilai perbandingan seperti dalam tabel V.

Pada tabel V *software* I dibandingkan *software* II adalah a/b karena kualitas berbanding terbalik dengan nilai parameter sehingga harus diinversi menjadi 1/(a/b). Sedangkan untuk perbandingan *software* II dan *software* I adalah kebalikan dari 1/(a/b) menjadi 1/(1/(a/b)). Kaidah yang sama diterapkan pada nilai-nilai perbandingan yang lain.

Tahap pembobotan properti kualitas desain *software* secara umum sama prosedurnya dengan pembobotan parameter MOOSE CK. Perbedaannya terletak pada objek yang dibandingkan, yaitu objek parameter MOOSE CK dan properti kualitas desain *software*. Untuk memperjelas keseluruhan metode pengukuran kualitas desain *software*, peneliti menjelaskan proses pembobotan properti kualitas desain *software*. Detail proses pembobotan properti kualitas desain *software* seperti dalam [13].

TABEL V
PERBANDINGAN *SOFTWARE* BERDASARKAN WMC

| WMC | S/W I | S/W II | S/W III |
|---|---|---|---|
| S/W I | 1 | 1/(a/b) | 1/(a/c) |
| S/W II | 1/(1/(a/b)) | 1 | 1/(b/c) |
| S/W III | 1/(1/(a/c)) | 1/(1/(b/c)) | 1 |

Perbandingan Nilai Skala Saaty pada Properti Kualitas Desain *Software*: Sama seperti prosedur pembobotan parameter MOOSE CK, mula-mula perbandingan pada masing-masing properti kualitas desain *software* dengan menggunakan skala Saaty. Kembali peneliti tekankan bahwa dalam perbandingan kualitas desain *software*, seorang pengukur harus menentukan terlebih dahulu objektifitas pengukuran kualitas desain *software*. Dalam contoh ini peneliti lebih menekankan pada properti kualitas dengan urutan sebagai berikut: *maintanability/testability, reusability, understand-ability, efficiency*. Sehingga hasil dari perbandingan tersebut seperti pada tabel VI.

Dengan cara yang sama dapat dihitung *consistency vector* dan *consistency ratio* yang dihasilkan dari perbandingan antar properti. Hasilnya adalah CI = 0.0249 dan dapat diterima.

TABEL VI
PERBANDINGAN PROPERTI KUALITAS DESAIN

| | Effic iency | Underst andabilit y | Reusabil ity | Maintaina bility/ Testabilty |
|---|---|---|---|---|
| Efficiency | 1.00 | 0.25 | 0.17 | 0.14 |
| Understand ability | 4.00 | 1.00 | 0.50 | 0.25 |
| Reusability | 6.00 | 2.00 | 1.00 | 0.50 |
| Maintainab ility/ Testabilty | 7.00 | 4.00 | 2.00 | 1.00 |

Perhitungan *Lambda* menggunakan hasil dari tabel VI. *Consistency Vector* diperoleh dengan menjumlahkan semua *consistency vector* dibagi jumlah *item* yang dievaluasi seperti pada persamaan (4.5). Dalam kasus ini jumlah *item* = 4.

$$\lambda = \frac{4.0144 + 4.0552 + 4.0802 + 4.192}{4} = 4.0673 \qquad (4)$$

$$CI = \frac{\lambda - n}{n - 1} = \frac{4.0673 - 1}{4 - 1} = 0.02242 \qquad (5)$$

$$CR = \frac{CI}{RI} = \frac{0.02242}{0.9} = 0.0249 \qquad (6)$$

Evaluasi Properti Kualitas Desain: Evaluasi properti kualitas desain *software* merupakan tahapan terakhir dari empat tahapan utama metode pengukuran kualitas desain *software*. Pada tahap ini menggunakan hasil dari tiga tahapan sebelumnya yaitu hasil faktor evaluasi MOOSE CK, hasil faktor pembobotan MOOSE CK dan hasil faktor pembobotan properti kualitas desain *software*. dan hasil akhirnya adalah final evaluasi dari keseluruhan *software*.

Ada dua tahapan lagi dalam evaluasi properti kualitas desain *software* yaitu mnghitung faktor evaluasi sesuai dengan properti kualitas desain *software* dan menghitung final evaluasi masing-masing *software*.

Menghitung Faktor Evaluasi Sesuai dengan Properti Kualitas Desain *Software*: Pada tahapan ini, faktor pembobotan MOOSE pada masing-masing *software* (tabel IV) dan hasil faktor evaluasi MOOSE CK (tabel VII) digunakan untuk menentukan nilai masing-masing properti. Masing-masing parameter MOOSE CK memiliki kontribusi yang berbeda-beda dengan properti kualitas *software*. Perhitungan masing-masing properti kualitas dilakukan dengan formula di tabel VIII.

TABEL VII
Hasil Perhitungan Bobot Properti

| Faktor | Bobot |
|---|---|
| Efficiency | 0.0527 |
| Understandability | 0.1571 |
| Reusability | 0.2865 |
| Maintainability / Testability | 0.5036 |

Faktor Evaluasi (FE) dan Bobot *Metric* (BM) disesuaikan dengan nilai pada *item software* yang dipilih. Perhitungan final evaluasi mengalikan antara bobot properti kualitas *software* (tabel VII) dengan hasil perhitungan faktor evaluasi dari properti kualitas di tabel VIII. Faktor Evaluasi (FE) disesuaikan dengan *item* yang berhubungan yaitu *software* I/II/III**.**

TABEL VIII
PERHITUNGAN FAKTOR EVALUASI PADA PROPERTI KUALITAS DESAIN *SOFTWARE*

| Properti | *Software* I/II/III |
|---|---|
| Efficy | $(FE)_{DIT}\ (BM)_{DIT} + (FE)_{NOC}\ (BM)_{NOC} + (FE)_{CBO}\ (BM)_{CBO} + (FE)_{LCOM}\ (BM)_{LCOM}$ |
| Undstbty | $(FE)_{WMC}\ (BM)_{WMC} + (FE)_{DIT}\ (BM)_{DIT} + (FE)_{RFC}\ (BM)_{RFC}$ |
| Reusbty | $(FE)_{WMC}\ (BM)_{WMC} + (FE)_{DIT}\ (BM)_{DIT} + (FE)_{NOC}\ (BM)_{NOC} + (FE)_{CBO}\ (BM)_{CBO} + (FE)_{LCOM}\ (BM)_{LCOM}$ |
| Mntnbty/ Testbty | $(FE)_{WMC}\ (BM)_{WMC} + (FE)_{DIT}\ (BM)_{DIT} + (FE)_{NOC}\ (BM)_{NOC} + (FE)_{RFC}\ (BM)_{RFC} + (FE)_{LCOM}\ (BM)_{LCOM}$ |

Nilai akhir yang dipakai sebagai acuan dalam menentukan peringkat kualitas *software* dihasilkan oleh persamaan di bawah ini.

$$\begin{aligned} Software\ I/II/III = {} & (FE)_{efficiency} \times (BK)_{efficiency} + \\ & (FE)_{Understandability} \times (BK)_{Understandability} \\ & + (FE)_{Reusability} \times (BK)_{Reusability} + \\ & (FE)_{Testability} \times (BK)_{Testability} + \\ & (FE)_{Maintanability} \times (BK)_{Maintanability} \end{aligned} \qquad (7)$$

## 3. Hasil dan Pembahasan

Dengan menggunakan CKJM 1.8, hasil pengukuran nilai *metrics* MOOSE dari kelima aplikasi ERP ditampilkan dalam tabel IX. Nilai ini diolah oleh proses AHP dengan bobot *metrics* dan properti yang telah ditentukan sebelumnya, yaitu pada tabel IV dan tabel VII.

TABEL IX
NILAI MOOSE APLIKASI

| Faktor | Admp | Opnbrv | Plzm | Frdm | JAIO |
|---|---|---|---|---|---|
| WMC | 14.63 | 11.47 | 9.55 | 14.26 | 57.62 |
| DIT | 0.86 | 0.80 | 0.71 | 0.90 | 3.40 |
| NOC | 0.59 | 0.53 | 0.41 | 0.72 | 2.27 |
| CBO | 4.47 | 4.97 | 5.06 | 5.03 | 17.65 |
| RFC | 27.29 | 23.33 | 22.82 | 28.85 | 107.54 |
| LCOM | 129.03 | 115.68 | 135.82 | 130.23 | 512.35 |

CKJM 1.8 juga menghitung jumlah *class* yang terlibat dalam proses internal aplikasi ketika dijalankan secara operasional. Jumlah *class* dari masing-masing aplikasi ERP ditampilkan pada tabel X.

TABEL X
JUMLAH *CLASS* DALAM APLIKASI

| Aplikasi | Jumlah *class* |
|---|---|
| Adempiere 3.2 | 16206 |
| OpenBravo 2.5 | 13931 |
| Plazma 0.1.5 | 14489 |
| FreeDomERP 1.1.3.5 | 5089 |
| JAllInOne 0.9.0.6 | 5503 |

Dengan memasukkan nilai-nilai pengukuran dan pembobotan ke dalam formula di tabel VIII, diperoleh nilai faktor evaluasi seperti dalam tabel XI.

TABEL XI
HASIL PERHITUNGAN FAKTOR EVALUASI WMC UNTUK MASING-MASING APLIKASI

| Aplikasi | Faktor evaluasi |
|---|---|
| Adempiere | 0.1967 |
| Openbravo | 0.2509 |
| Plazma | 0.3023 |
| FreedomERP | 0.2018 |
| JAllInOne | 0.0499 |

Nilai akhir yang menentukan peringkat kualitas aplikasi ditentukan dengan persamaan 7. Dengan informasi nilai yang sudah diketahui, dapat diperoleh nilai akhir aplikasi ERP seperti dalan tabel XII. Hasil eksperimen menunjukkan dari aspek orientasi objek dengan menggunakan MOOSE CK, *software* ERP Plazma memiliki kualitas paling baik dibandingkan aplikasi ERP lain.

TABEL XII
URUTAN PERINGKAT KUALITAS

| Peringkat | Aplikasi | Nilai akhir |
|---|---|---|
| 1 | Plazma | 0.3023 |
| 2 | Openbravo | 0.2509 |
| 3 | FreedomERP | 0.2018 |
| 4 | Adempiere | 0.1967 |
| 5 | JAllInOne | 0.0499 |

## 4. Kesimpulan

Dalam penelitian ini telah diusulkan sebuah metode baru untuk menentukan peringkat kualitas desain secara relatif terhadap sekumpulan *software* berdasarkan implementasi *code*-nya dalam Java. Metode ini merupakan kombinasi dari konsep AHP, *object oriented metrics* dan properti kualitas.

Metode baru tersebut telah diujicobakan ke lima aplikasi ERP berbasis Java, yaitu Adempiere 3.2., OpenBravo 2.5., Plazma 0.1.5., FreedomERP 1.1.3.5., dan JAllInOne 0.9.0.6. Kelimanya dipilih sebagai *sample* oleh karena popularitas dan jumlah pemakainya tertinggi menurut versi SourceForge. Hasil pengukuran dan pengolahan informasi dengan AHP menghasilkan peringkat kualitas sebagaimana tersaji di tabel XII. Nilai akhir yang diperoleh menginterpretasikan kualitas desain *software* relatif terhadap *software* yang lain. Metode ini bersifat umum dan dapat diterapkan untuk mengevaluasi semua tipe *software* seperti: aplikasi desktop, aplikasi web, *class library*, atau API. Tetapi *software* yang akan dievaluasi seharusnya memiliki domain atau fungsi yang sama.

Metode ini juga dapat dipergunakan untuk mengukur *software* selain berbasis bahasa pemrograman Java, tentunya dengan alat ukur *metrics* yang sesuai dengan bahasa pemrograman yang dipakai. Metode ini bersifat fleksibel untuk pengukuran peringkat kualitas berdasarkan kombinasi properti yang bervariasi. Penentuan bobot properti dapat disesuaikan sehingga menggambarkan properti kualitas yang hendak diutamakan dalam evaluasi dan relatif terhadap properti evaluasi yang lain.

Pemakai metode ini selayaknya memahami arti fisis masing-masing parameter MOOSE CK, properti kualitas desain dan objektifitas dari pengukuran *software*. Objektivitas dari pengukuran dicapai dengan menerapkan pembobotan *property* maupun *metrics* yang sama dalam AHP dan diterapkan terhadap sejumlah aplikasi yang dievaluasi.

Riset lanjutan perlu dilakukan untuk penerapan metode ini dengan menggunakan himpunan *metric* yang lain, misalnya *Metrics for Object Oriented Design* (MOOD) [2] ataupun MOOD2 [1]. Konsistensi peringkat kualitas aplikasi dapat dievaluasi dengan membandingkan hasil penerapan metode ini menggunakan beberapa himpunan *metric*. Validasi pembobotan *metric* dan *property* pun dapat dievaluasi dengan mengamati konsistensinya ketika diterapkan dengan berbagai variasi parameter-parameter yang terlibat.

**Referensi**


[1] F. Brito e Abreu, The MOOD2 metrics set (in portuguese); relatorio r7/98, Technical report, Grupo de Engenharia de Software, INESC, 1998.

[2] F. Brito e Abreu, Talk on "Design Metrics for Object-Oriented Software Systems", *In ECOOP'95 Quantitative Methods Workshop*, Aarhus, 1995, http://ctp.di.fct.unl.pt/QUASAR/ECOOP95/ecoop95_submission.pdf, retrieved March 24, 2009.

[3] S.R. Chidamber & C.F. Kemerer, "MOOSE: Metrics for Object Oriented Software Engineering" *Workshop on Processes and Metrics for Object-Oriented Software Development (OOPSLA'93)*, pp. 554-560, 1993.

[4] L. H. Rosenberg & L.E. Hyaat, "Software Quality Metrics for Object-Oriented Environments" *Unisys Technology Conference*, pp. 1-6, 2003.

[5] J. Lindell & M. Hägglund, Maintainability Metrics for Object Oriented Systems, 2004.

[6] M. Bruntink & A.V. Deursen, "Predicting Class Testability using Object-Oriented Metrics" *Proceedings of the, 4th IEEE International Workshop on Source Code Analysis and Manipulation,* pp. 136-145, 2004.

[7] D. Spinellis, CKJM 1.8, www.spinellis.gr/sw/ckjm/, 2006, retrieved March 1, 2009.

[8] S.R. Chidamber & C.F. Kemerer, "A Metrics Suite for Object Oriented Design" M.I.T. Sloan School of Management, http://web.cs.wpi.edu/~gpollice/cs562-s05/Readings/CKMetrics.pdf, 1995, retrieved March 4, 2009.

[9] J. McCall, P. Richards, & G. Walters, *Factors in Software Quality*, 1977.

[10] B.W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, & M. J. Merritt, *Characteristics of Software Quality*, North Holland Publishing Company, Amsterdam, 1978.

[11] Thomas Saaty, *The Analytic Hierarchy Process*, McGraw Hill Book Company, New York, 1980.

[12] S.R. Chidamber & C.F. Kemerer, "*Towards a Metrics Suite for Object Oriented Design*" *In Proceeding Sixth OOPSLA Conference*, pp. 197-211, 1991.

[13] E. Hermawan, "Pemeringkatan Kualitas Desain Software Berdasarkan MOOSE dengan Mempergunakan Analytic Hierarchy Process," Ph.D Thesis, Magister of Information Technology, Universitas Indonesia, 2008.