USULAN MODEL MENGUKUR KOMPLEKSITAS KODE PROGRAM PADA PERANGKAT LUNAK

Gede Herdian Setiawan[1], I Made Budi Adnyanan[2], Gusti Ngurah Mega Nata[3]

[123,]Fakultas Informatika dan Komputer Institut Teknologi dan Bisnis STIKOM Bali

Jln. Raya Puputan Renon No. 86 Renon Denpasar Bali
[1] herdian@stikom-bali.ac.id, [2]budi.adnyana.stikom-bali.ac.id, [3]mega.stikom-bali.ac.id

Abstract

The quality of writing source code is one of the most important *things in getting good application performance, writing complex source code and not paying attention to attributes effectively and efficiently affects the performance of a software. Measuring the level of complexity of writing code in software can assist developers in planning and controlling software development in order to provide the expected performance. This study intends to propose a model to measure the complexity of writing program code in software by taking into account the quantity and quality of each attribute used by giving weight values to each attribute. The stages carried out in this research are determining which module to be analyzed in the software then dividing it into several groups of attributes in each module, then giving weight to each attribute, then calculating the weight value for each attribute in the program code and analyzing the complexity of the program code. Based on the results of the complexity calculation, the value or number of the complexity level of the program code in each module shows the average complexity value for the three modules, namely 3.03. Thus, the software measured using the proposed model has a very low complexity rating.*

Keywords*: Software, Source Code, Complexity, Measuring*

Abstrak

Kualitas penulisan kode program menjadi salah satu hal yang terpenting dalam mendapatkan performa aplikasi yang baik, penulisan kode program yang komplek dan tidak memperhatikan atribut secara efektif dan efesien berpengaruh pada performa sebuah perangkat lunak. Mengukur tingkat kompleksitas penulisan kode pada perangkat lunak dapat membantu pengembang dalam melakukan perencanaan dan pengendalian pengembangan perangkat lunak agar dapat memberikan performa sesuai dengan yang diharapkan. Penelitian ini bermaksud untuk mengusulkan model untuk mengukur kompleksitas penulisan kode program pada perangkat lunak dengan memperhatikan kuantitas dan kualitas setiap atribut yang digunakan dengan memberikan nilai bobot pada setiap atributnya. Tahapan yang dilakukan pada peneilitian ini yaitu menentukan modul yang dianalisa pada perangkat lunak selanjutnya membagi menjadi beberapa kelompok atribut pada masing-masing modul, kemudian memberikan bobot pada setiap atributnya, selanjutnya menghitung nilai bobot pada setiap atribut pada kode program dan melakukan analisa terhadap kompleksitas kode program. Berdasarkan hasil perhitungan kompleksitas telah dihasilkan nilai atau angka tingkat kompleksitas kode program pada setiap modul menunjukkan nilai rata-rata kompleksitas pada ketiga modul yaitu 3,03. Dengan demikian perangkat lunak yang diukur menggunakan model yang diusulkan memiliki *rating complexity Very low.*

Kata kunci *: Perangkat Lunak, Kode Program, Kompleksitas, Mengukur*

1. PENDAHULUAN

Aplikasi (Perangakat lunak) pengamatan kondisi lalu lintas merupakan aplikasi yang dibangun untuk memberikan informasi kondisi lalu lintas secara instan, cepat dan akurat. Perangkat lunak dibangun dengan dua bahasa pemrograman yaitu pada server menggunakan bahasa pemrograman PHP dan pada *client* menggunakan bahasa pemrogramana Java. *server* bertugas untuk menyimpan dan memproses data, sedangkan *client* untuk mengambil data dan menampilkan informasi kepada pengguna. [1][2]

Desain perangkat lunak yang cukup komplek dan arus pemindaian data yang cepat bisa mengakibatkan ketidakmampuan perangkat lunak dalam mengelola data dan informasi ketika terjadi pemrosesan data yang banyak seperti pada aplikasi datamining sehingga dapat mempengaruhi performa pada perangkat lunak. terdapat berbagai macam hal yang harus diperhatikan untuk mendapatkan performa perangkat lunak yang baik salah satunya adalah penulisan kode program. Kualitas penulisan kode program menjadi salah satu hal yang terpenting dalam mendapatkan performa aplikasi yang baik, penulisan kode program yang kompleks dan tidak memperhatikan atribut secara efektif dan efesien berpengaruh pada performa sebuah perangkat lunak.

Mengukur tingkat kompleksitas penulisan kode pada perangkat lunak dapat membantu pengembang dalam melakukan perencanaan dan pengendalian pengembangan perangkat lunak agar dapat memberikan performa sesuai dengan yang diharapkan.[3][4]

Terdapat beberapa metode yang digunakan untuk mengukur kompleksitas perangkat lunak berdasarkan penulisan kode program diantaranya metode *loc* dengan memperhatikan jumlah baris pada kode program. *Cyclomatic complexity* dengan memperhatikan jumlah kompleksitas logika atau kondisional pada penulisan kode program pada perangkat lunak dan metode *Halstead complexity* melakukan perhitungan dengan memperhatikan atribut seperti operator dan operand kode program pada sebuah perangkat lunak.[5] [6][7].

Dari penelitian yang sudah dilakukan tersebut metode yang digunakan dalam mengukur kompleksitas penulisan kode program lebih memperhatikan kuantitas (baris, logika dan atribut) kode program pada perangkat lunak tanpa memperhitungkan kesulitan atau kualitas pada suatu kode program. Melalui penelitian ini bermaksud untuk mengusulkan model untuk mengukur kompleksitas penulisan kode program dengan tetap memperhatikan kuantitas baris, logika dan atribut pada kode program namun juga mengukur kualitas setiap atribut kode program dengan memberikan nilai bobot pada setiap atribut pengujian yang melibatkan *expert* melalui wawancara.

2. TINJAUAN PUSTAKA DAN TEORI

2.1 Penelitian terkait

Mengukur tingkat kompleksitas penulisan kode pada perangkat lunak dapat membantu pengembang dalam melakukan perencanaan dan pengendalian pengembangan perangkat lunak agar dapat memberikan performa sesuai dengan yang diharapkan.

Pada penelitian terdahulu Wijendra, Dinuka Rukshani menggunakan perhitungan dengan metode *loc* dilakukan hanya memperhatikan jumlah baris pada kode program namun tidak memperhatikan tingkat kesulitan logika pada kode program.[5] [6]

G. K. Gill dan C. F. Kemerer melakukan perhitungan dengan *Cyclomatic complexity* dengan memperhatikan jumlah kompleksitas logika pada penulisan kode pada perangkat lunak. [8]

Sedangkan Hariprasad, Seenu dan Vidhyagaran menggunakan metode *Halstead complexity* untuk melakukan perhitungan dengan memperhatikan atribut seperti operator dan operand pada sebuah perangkat lunak.[7]

Selain memperhatikan atribut dalam penulisan kode program pada perangkat lunak, pengujian perangkat lunak pernah dilakukan Subali dan Rochimah dengan mengukur kompleksitas perangkat lunak berdasarkan atribut pada sistem database.[9]

2.2 Aplikasi Pengamatan Kondisi Lalu Lintas

Aplikasi pengamatan lalu lintas merupakan aplikasi yang dibangun untuk memberikan perakiraan kondisi lalu lintas pada ruas jalan. Perakiraan Kondisi lalu lintas berdasarkan data kecepatan kendaraan yang di rekam menggunakan perangkat *smartphone* dengan dengan memanfaatkan sensor akselerometer dan modul GPS. Kondisi lalu lintas di sajikan dengan menggunakan fitur garis berwarna *(polyline)* di *google maps* pada *smartphone* Andorid, yang menunjukkan kondisi lalu lintas macet pada garis berwarna hitam,kondisi lalu lintas padat garis berwarna merah, kondisi lalu lintas sedang garis berwarna kuning, kondisi lalu lintas lancar garis berwarna hijau. [2][10][11].

2.3 Program dan Bahasa Pemrograman

Program adalah algoritma yang ditulis dalam bahasa komputer. Pemrograman adalah proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengan menggunakan bahasa pemrograman. Penulisan program biasanya menggunakan menggunakan program editor yang telah disediakan oleh bahasa pemrograman yang dipilih.

Langkah-Langkah Pembuatan Program Dalam menyusun suatu program yang besar dan kompleks dibutuhkan beberapa tahapan yang sistematis dan terpadu, yaitu sebagai berikut:

1. Mendifinisikan masalah, Analisis kebutuhan, Desain algoritma, Pengkodean, Bahasa pemrograman, Testing dan debugging, Dokumentasi, Pemeliharaan.
2. Menentukan modul-modul program. Penentuan modul utama dan modul ritun sangat berguna untuk mempermudah penanganan jika terjadi kesalahan.
3. Penyusunan algoritma program. Algoritma dibuat dengan tujuan untuk menyelesaikan masalah dengan menuliskan langkah-langkah pemecahan masalah yang ada.

Bahasa pemrograman merupakan prosedur penulisan. Ada tiga record dalam penulisan bahasa pemograman:

1) Syntax adalah aturan penulisan bahasa tersebut (tata bahasa),
2) Semantic adalah arti atau maksud yang terkandung di dalam statement tersebut,
3) Kebenaran logika adalah berhubungan dengan benar tidaknya urutan statement. [12]

2.4 Komponen Bahasa pemrograman

Bahasa pemrograman terdiri beberapa notasi/statement pembentuk algoritma. Notasi/Statement pembentuk program dibungkus dalam sebuah procedure/function. Kompleksitas sebuah fungsi tergantung dari banyaknya notasi-notasi tertentu yang membentuk fungsi tersebut. Berikut adalah notasi pembentuk fungsi dari sebuah program.

Judul Algoritma {nama dari fungsi }

Deklarasi / kamus

- Deklarasi variable
  penggunaan alamat memori untuk menyimpan *value*
- Deklarasi konstanta
  penggunaan alamat memori untuk menyimpan *value* yang tidak bisa dirubah
- Deklarasi fungsi
  prototype dari fungsi yang disimpan dalam memori
- Deklarasi objek
  Memperkenalkan variable objek yang dibangun dari *class* objek. *Variable* objek tersimpan di memori

Tubuh Algoritma / Deskripsi

- Notasi Assigment
  Notasi yang menganti nilai pada sebuah *variable*
- Notasi kondisional / pemilihan
  Notasi ini memiliki kondisi untuk memecah alir program.
- Notasi Pengulangan
  Notasi ini memiliki kondisi yang diproses secara berulang selama kondisinya terpenguhi.
- Notasi pemanggilan
  Notasi ini melakukan pemanggilan (*call*) dari fungsi atau objek yang telah dideklarasikan.

Deklarasi / kamus adalah seperangkat statement / ekpresi yang diperkenalkan (*Allocation*) dan memerlukan ruang memori. Proses deklarasi tidak terlalu berat dibandingkan dengan proses perhitungan karena deklarasi hanya menggunakan ruang *memory* sebagai media penyimpanan dari *values* dengan typedata yang dimiliki. Namun, deklarasi tetap menjadi beban di ruang *memory* selama di alokasikan (*allocation*) karena memblok alamat memory yang tidak bisa digunakan lagi oleh deklarasi lain selama tidak di Dealokasi (*Deallocation*).

Sedangkan tubuh algoritma yang dibentuk dari notasi – notasi merupakan ekpresi yang memproses beberapa variable dengan cara membaca alamat dari alokasi variable kemudian melakukan perhitungan dan menyimpan hasil perhitungan pada alokasi *memory*. Kompleksitas notasi dapat dilihat dari berapa banyak alokasi memori yang terlibat dalam proses perhitungan algoritma tersebut.

2.5 Perangkat Lunak

Perangkat Lunak merupakan seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program maupun prosedur yang didalam nya merupakan kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi. [13]

Perangkat lunak pada komputer telah mengalami perkembangan yang sangat maju, berkembang mengikuti kebutuhan jaman. Mulai dari bentuk paling primitif dari perangkat lunak, menggunakan aljabar Boolean, yang di representasikan sebagai binary digit (bit), yaitu 1 (benar/on) atau 0 (salah/off), namun cara ini sudah pasti sangat menyulitkan, sehingga mulai dikelompokkan bit tersebut menjadi nible (4 bit), byte (8 bit), word (2 byte), double word (32 bit) sampai kepada bahasa pemrograman. Penggunaan *memory* pada perangkat lunak sangat berpengaruh terhadap kecepatan sehingga pemilihan tipe data berpengaruh terhadap kompleksitas.

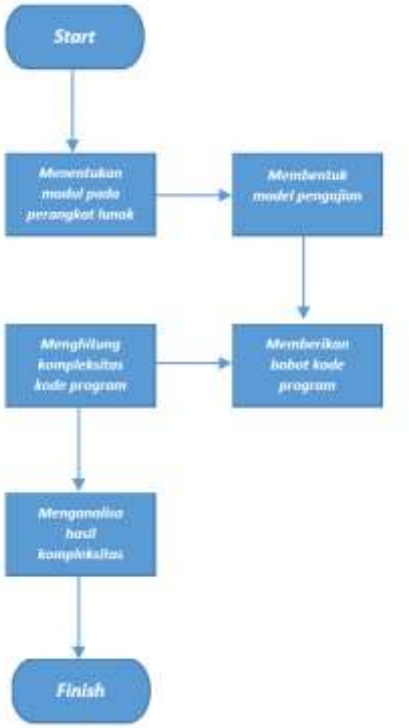
2.6 Kompleksitas Kode Program

Kompleksitas kode program dapat dibagi menjadi dua yaitu kompleksitas waktu proses dan kompleksitas ruang memory. Kompleksitas waktu merupakan kompleksitas yang disebabkan oleh banyaknya notasi – notasi yang digunakan dalam membentuk algoritma. Notasi pembentuk algoritma yang menyebabkan kompleksitas waktu yaitu notasi assigment, notasi kondisional / pemilihan, notasi pengulangan dan notasi pemanggilan. Kompleksitas ruang memory merupakan kompleksitas penggunaan alamat memori untuk menyimpan value dari deklarasi variable. [14]

3. METODOLOGI PENELITIAN

3.1 Skema Alur Penelitian

Penelitian ini dilakukan melalui beberapa tahapan yaitu : Tahap pertama menentukan unit atau modul pada perangkat lunak (Aplikasi Pengamatan Kondisi Lalu Lintas) yang akan dihitung tingkat kompleksitasnya dalam hal ini dilakukan dengan memperhatikan modul yang memiliki fungsi cukup penting pada perangkat lunak, tahap kedua membentuk model pengujian kode program pada setiap modul, tahap ketiga memberikan bobot pada setiap kode program yang diujikan, tahap selanjutnya menghitung kompleksitas kode program dan menganalisa hasil kompleksitas pada setiap modul yang diujikan. Tahapan penelitian ditunjukkan pada Gambar 1



Gambar 1. Tahapan Penelitian

3.2 Menentukan Modul Uji

Penentuan modul dengan mempertimbangkan fitalitas fungsi pada modul tersebut yang akan mempengaruhi performa pada perangkat lunak. Dalam penelitian ini menggunakan modul pengambilan data pada sisi *client* dan modul penyajian data pada sisi *server*.

3.3 Membentuk Model Pengujian

Setelah modul pada perangkat lunak ditentukan, selanjutnya dilakukan pengelompokan atribut pada masing-masing modul. Pengelompokan atribut pada kode program dibagi beberapa jenis : Kondisi, Perulangan, Fungsi, Operand dan Operator.

3.4 Memberikan Bobot Pada Atribut

Setelah pengelompokan sudah terbentuk selanjutnya dilalukan pemberian bobot pada setiap atribut yang dinilai. Pada tahapan ini penulis melibatkan *expert* dalam melakukan penilaian kualitas pada setiap atribut yang diujikan. Pemberian bobot menggunakan skala penilaian dari rentan 0 sampai 1, 0 merupakan nilai terendah yang memiliki bobot paling kecil sementara 1 nilai tertinggi

yang memiliki bobot paling tinggi dan memiliki pengaruh terhadap kualitas kode program.

3.5 Menghitung Kompleksitas

Pada tahapan ini dilakukan proses penghitungan kompleksitas kode program dengan memperhatikan informasi jumlah pada setiap atribut pada modul beserta nilai bobotnya. Rumus untuk menghitung kompleksitas kode program sebagai berikut:[6]

$$KKP = \sum_{i=1}^{n} x_i \, w_i \qquad (1)$$

dimana:
n merupakan total atribut pada kode program
x_i merupakan nilai atribut pada kode program ke *i*
w_i merupakan bobot atribut pada kode program ke *i*

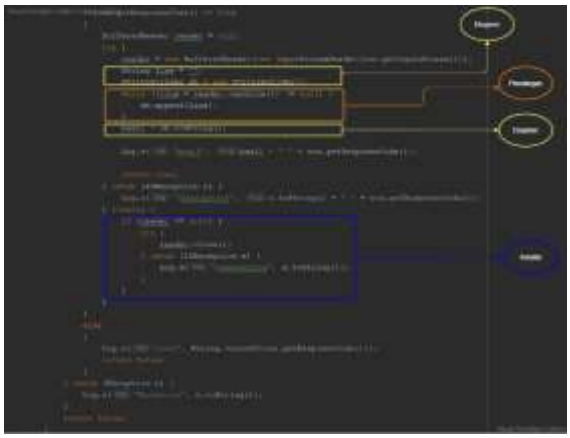
3.6 Analisa Kompleksitas

Pada tahapan ini memberikan rangkuman terhadap hasil dari proses pengukuran kompleksitas kode program sekaligus melakukan evaluasi model dan pemilihan pengelompokan atribut.

4. HASIL DAN PEMBAHASAN

4.1 Menentukan Modul Perangkat Lunak

Penentuan modul dengan mempertimbangkan fitalitas fungsi pada modul tersebut yang diyakini akan mempengaruhi performa pada perangkat lunak. Dalam penelitian ini menggunakan modul pengambilan data pada sisi *client*, modul penyimpanan data pada sisi *server* dan modul menampilkan informasi pada *client user*.

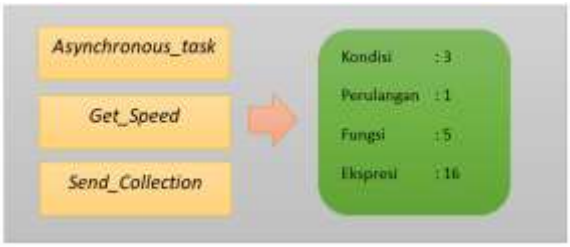


Gambar 2. Menentukan Modul Perangkat Lunak

Pada setiap modul yang sudah ditentukan kemudian dilakukan ekstraksi atribut pada kode program. Gambar 2 menunjukkan ekstrakasi atribut pada salah satu modul perangkat lunak dengan mencari beberapa jenis atribut seperti : Kondisi, Perulangan, Fungsi dan Ekspresi.

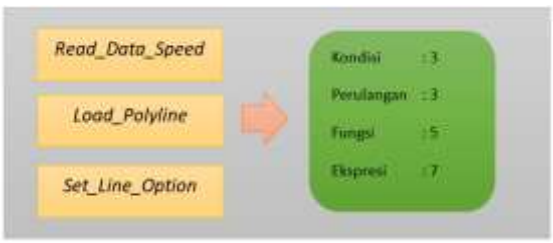
4.2 Membentuk Model Pengujian

Untuk membetuk model pengujian pada kode program dilakukan pengelompokan atribut pada masing-masing modul setelah dilakukan proses ekstraksi. Pengelompokan atribut pada kode program dibagi beberapa jenis : Kondisi, Perulangan, Fungsi dan Ekspresi. Setiap modul dibagi menjadi beberapa sub sesuai dengan class atau fungsi pada kode program.



Gambar 2. Model Pengujian Pengambilan Data

Gambar 3. Model Pengujian Pemrosesan Data



Gambar 4. Model Pengujian Menampilkan Data

4.3 Memberikan Bobot Pada Kode Program

Setelah pengelompokan sudah terbentuk selanjutnya dilalukan peberian bobot pada setiap atribut yang dinilai. Terdapat empat atribut kode program yang dinilai, antara lain: Kondisi, Perulangan, Fungsi dan Ekspresi.

Kondisi merupakan atribut yang berfungsi untuk memberikan logika pada baris program sebagai langkah untuk mengeksekusi perintah – perintah pada baris program. Perulangan merupakan atribut yang berfungsi sebagai sebuah logika yang akan melakukan eksekusi perintah pada baris program secara berulang. Fungsi merupakan

suatu blok kode yang akan mengeksekusi perintah-perintah ketika dipanggil dari bagaian lain dalam baris program. Ekspresi merupakan baris perintah untuk memberikan atau memasukkan nilai ke dalam variabel. Pada tabel 1 merupakan informasi nilai bobot yang diusulkan pada setiap atribut kode program.

Tabel 1. Nilai Bobot Atribut Kode Program

No	*Atribut (x)*	*Bobot (w)*
1	Kondisi	0,30
2	Perulangan	0,30
3	Fungsi	0,20
4	Ekspresi	0,10

4.4 Menghitung Kompleksitas Kode Program

Pada tahapan ini dilakukan proses penghitungan kompleksitas kode program dengan memperhatikan informasi jumlah pada setiap atribut pada modul beserta nilai bobotnya. Rumus yang digunakan seperti ditunjukkan pada persamaan 1.

Tabel 2. Kompleksitas Kode Program Modul Pengambilan Data

No	*Atribut (x)*	*Bobot (w)*	x_i	$x_i . w_i$
1	Kondisi	0,30	3	0,9
2	Perulangan	0,30	1	0,3
3	Fungsi	0,20	5	1
4	Ekspresi	0,10	16	1,6
∑ Kompleksitas				**4,80**

Tabel 2 menunjukkan kompleksitas kode program pada modul pengambilan data dengan nilai kompleksitas yaitu 4,80 dengan bobot atribut eksepresi memiliki nilai paling tinggi yaitu 1,6

Tabel 3. Kompleksitas Kode Program Modul Penyimpanan Data

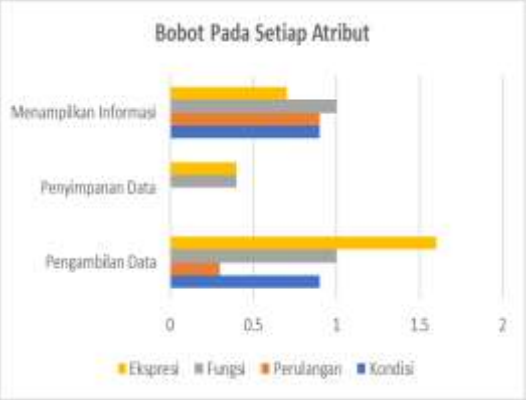
No	*Atribut (x)*	*Bobot (w)*	x_i	$x_i . w_i$
1	Kondisi	0,30	0	0
2	Perulangan	0,30	0	0
3	Fungsi	0,20	2	0.4
4	Ekspresi	0,10	4	0.4
∑ Kompleksitas				**0.8**

Tabel 3 menunjukkan kompleksitas kode program pada modul penyimpanan data dengan nilai kompleksitas yaitu 0,8 dengan bobot atribut hanya pada atribut fungsi dan eksepresi yang memilki masing-masing nilai yaitu 0,4

Tabel 4. Kompleksitas Kode Program Modul Menampilkan Informasi

No	*Atribut (x)*	*Bobot (w)*	x_i	$x_i . w_i$
1	Kondisi	0,30	3	0.9
2	Perulangan	0,30	3	0.9
3	Fungsi	0,20	5	1
4	Ekspresi	0,10	7	0.7
∑ Kompleksitas				**3,5**

Tabel 4 menunjukkan kompleksitas kode program pada modul menampilkan informasi dengan nilai kompleksitas yaitu 3,5 dengan bobot nilai yang palng besar pada atribut kondisi dan perulangan yaitu 0,9



Gambar 4. Grafik Bobot Pada Setiap Atribut

Gambar 4 menunjukkan secara grafis nilai bobot pada setiap atribut pada ketiga modul, nilai atribut ekspresi pada modul pengambilan data sangat tinggi dengan nilai bobot yaitu 1,6.

4.5 Hasil Kompleksitas Kode Program

Hasil kompleksitas kode program pada 3 modul yang telah dihitung seperti ditunjukkan pada Tabel 2, Tabel 3 dan Tabel 4. Dimana nilai pada modul pengambilan data paling tinggi, diikuti pada modul menampilkan informasi sedangkan modul penyimpanan data memproleh nilai paling rendah, dalam bentuk grafik ditunjukkan pada Gambar 5.

Gambar 5. Grafik Kompleksitas Kode Program

Berdasarkan hasil perhitungan kompleksitas telah dihasilkan nilai atau angka tingkat kompleksitas kode program pada setiap modul. Namun belum dapat menerangkan secara lebih harfiah kompleksitas perangkat lunak, dengan menggunakan tabel *product complexity rating* seperti penelitian yang dilakukan Subandari dan Sarno dapat menerangkat secara lebih harfiah angka dari hasil perhitungan kompleksitas. *Product complexity rating* yang digunakan sebagai rujukan pada penelitian seperti ditunjukkan pada Tabel 5. Berdasarkan nilai rata-rata terhadap tiga modul yang diujikan memperoleh nilai rata-rata 3,03 dengan demikian rating pada kategori *very low*.[15]

Tabel 5. *Product Complexicty Rating*

No.	***Complexity***	***Rating***
1	1 – 4	*very low*
2	5 – 10	*low*
3	11 – 20	*normal*
4	21 – 40	*high*
5	41 – 50	*very high*
6	> 51	*extra high*

5. KESIMPULAN DAN SARAN

Telah diterapkan usulan model untuk mengukur kompleksitas penulisan kode program pada perangkat lunak dengan tidak hanya mengukur secara kuantitas namun juga mengukur kualitas setiap atribut yang digunakan dengan memberikan nilai bobot pada setiap atributnya yang melibatkan *expert*. Tahapan yang dilakukan dengan menentukan modul pada perangkat lunak yang diujikan, membentuk model pengujian, memberikan bobot pada kode program kemudian menghitung kompleksitas kode program.

Berdasarkan hasil pengukuran kompleksitas telah dihasilkan nilai atau angka tingkat kompleksitas kode program pada setiap modul menunjukkan nilai pada modul pengambilan data paling tinggi, diikuti pada modul menampilkan informasi sedangkan modul penyimpanan data memproleh nilai paling rendah dengan nilai rata-rata kompleksitas pada ketiga modul yaitu 3,03.

Dengan demikian kode program pada perangkat lunak yang diukur menggunakan model yang diusulkan memiliki rating *complexity Very low.*

DAFTAR PUSTAKA

[1] G. H. Setiawan and I. K. D. Suryawan, "Analisis Kinerja Aplikasi Pengamatan Kondisi Lalu Lintas Berdasarkan Tingkat Akurasi," pp. 579–583, 2020.

[2] G. H. Setiawan and I. K. D. Suryawan, "Sistem Pengamatan Kondisi Lalu Lintas Berbasis Data GPS pada Smartphone (Studi Kasus : Kota Denpasar)," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 4, p. 667, 2020, doi: 10.25126/jtiik.2020741909.

[3] G. N. M. Nata, "Association Rule Mining Pada Data Geokimia Pemboran," *Eksplora Inform.*, vol. 5, no. 2, pp. 131–136, 2016.

[4] G. Ngurah, M. Nata, and P. P. Yudiastra, "Knowledge discovery pada email box sebagai penunjang email marketing knowledge discovery in the email box for support email marketing," *J. Sist. dan Inform.*, pp. 26–37, 2017.

[5] D. R. Wijendra and K. P. Hewagamage, "Automated tool for the calculation of cognitive complexity of a software," *Proceeding - 2016 2nd Int. Conf. Sci. Inf. Technol. ICSITech 2016 Inf. Sci. Green Soc. Environ.*, pp. 163–168, 2017, doi: 10.1109/ICSITech.2016.7852627.

[6] S. Aswini and M. Yazhini, "An assessment framework of routing complexities using LOC metrics," *2017 Innov. Power Adv. Comput. Technol. i-PACT 2017*, vol. 2017-Janua, no. April, pp. 1–6, 2017, doi: 10.1109/IPACT.2017.8245022.

[7] T. Hariprasad, G. Vidhyagaran, K. Seenu, and C. Thirumalai, "Software complexity analysis using halstead metrics," *Proc. - Int. Conf. Trends Electron. Informatics, ICEI 2017*, vol. 2018-Janua, no. May, pp. 1109–1113, 2018, doi: 10.1109/ICOEI.2017.8300883.

[8] G. K. Gill and C. F. Kemerer, "Cyclomatic Complexity Density and Software Maintenance Productivity," *IEEE Trans. Softw. Eng.*, vol. 17, no. 12, pp. 1284–1288, 1991, doi: 10.1109/32.106988.

[9] M. A. P. Subali and S. Rochimah, "A new model for measuring the complexity of SQL commands," *Proc. 2018 10th Int. Conf. Inf. Technol. Electr. Eng. Smart Technol. Better Soc. ICITEE 2018*, pp. 1–5, 2018, doi: 10.1109/ICITEED.2018.8534782.

[10] E. Syahputra, I. B. K. Widiartha, and A. Zubaidi, "Rancang Bangun Sistem Informasi Geografis Pemetaan Daerah Rawan Kriminalitas Dikota Mataram Berbasis Web," *J. Manaj. Inform. dan Sist. Inf.*, vol. 2, no. 2, p. 39, 2019, doi: 10.36595/misi.v2i2.102.

[11] E. S. Han and A. goleman, daniel; boyatzis, Richard; Mckee, "済無No Title No Title," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2019.

[12] E. W. Fridayanthie and J. Charter, "ISSN 1978-2136 | Rancang Bangun Sistem ... ISSN 1978-2136 | Rancang Bangun Sistem ...," vol. XIII, no. 2, pp. 63–71, 2016.

[13] Y. Swara, G. Y. Kom. M., & Pebriadi, "Rekayasa Perangkat Lunak Pemesanan Tiket Bioskop Berbasis Web," *J. TEKNOIF*, vol. 4, no. 2, pp. 27–39, 2016.

[14] S. Ariani, B. Santosa, S. E. Wiratno, and R. Suprianti, "Analisis Kompleksitas Algoritma Biogeography Based Optimization (BBO) Pada Traveling Salesman Problem (TSP)," vol. 1, no. 1, pp. 10–17, 2020.

[15] M. A. Subandri and R. Sarno, "Cyclomatic Complexity for Determining Product Complexity Level in COCOMO II," *Procedia Comput. Sci.*, vol. 124, pp. 478–486, 2017, doi: 10.1016/j.procs.2017.12.180.