Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 5 Nomor 1 Februari, pp 147-153
ISSN: 2527-5771/EISSN: 2549-7839
http://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Pengukuran Metrik Kompleksitas *Web Service* Sekolah

Ahmad Riza[1], Mohamad Alif Irfan Anshori[2], Farrah Arrazy[3]
[1),2),3)] Teknik Informatika, Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Jl. Gajayana No. 50 Malang 65144 – Indonesia
Email: riza.public@gmail.com[1)], alif.anshory16@gmail.com[2)], farraharrazyrazy@gmail.com[3)]

Abstract

The Complexity Metric Diagram is a method of measuring software complexity. This method uses Class Diagrams as test parameters. This study uses test data from previous research regarding the design of Information Systems accompanied by Class Diagrams with case studies of School Academic Information Systems. The facts found in the field that school software developers do not conduct application audits so it is not ripe to face problems that will arise in the future. The results of the calculation of the complexity metrics in the School Academic Information System produce an output of 993.32 units.

Keywords: *Complexity Metrics, Web Service, School Academic System*

Abstrak

Metriks Kompleksitas Diagram merupakan metode pengukuran tingkat kerumitan perangkat lunak. Metode ini mengunakan Class Diagram sebagai parameter ujinya. Penelitian ini menggunakan data uji dari penelitian sebelumnya mengenai perancangan Sistem Informasi yang disertai Class Diagram dengan studi kasus Sistem Informasi Akademik Sekolah. Fakta dilapangan ditemukan bahwa pengembang perangkat lunak sekolah kurang melakukan audit aplikasi sehingga tidak matang menghadapi masalah yang akan muncul di masa yang akan datang. Hasil dari perhitungan metriks kompleksitas pada Sistem Informasi Akademik Sekolah menghasilkan output sebesar 993,32 unit.

Kata Kunci: *Metrik Kompleksitas, Web Service, Sistem Akademik Sekolah*

1. PENDAHULUAN

Pada zaman yang dimana teknologinya telah berkembang dengan pesatnya, banyak teknologi yang seringkali membantu kehidupan masyarakat sehari-hari, seperti *Internet of Things* (IoT) terutama *software* berbasis *mobile*, *website*, dan *desktop*. Perkembangan teknologi terutama dalam bidang sistem informasi saat ini sangat memungkinkan dalam pemanfaatan penyebaran informasi. Dalam hal ini, dapat dimanfaatkan di bidang pendidikan. Sebelumnya, menurut UU No 20 Tahun 2003, pendidikan adalah usaha sadar dan terencana untuk mewujudkan suasana belajar dan proses pembelajaran agar peserta didik secara aktif mengembangkan potensi dirinya untuk memiliki kekuatan spiritual keagamaan, pengendalian diri, kepribadian, kecerdasan, akhlak mulia, serta keterampilan yang diperlukan dirinya, masyarakat, bangsa, dan negara. Terkait hal itu, sistem informasi dapat membantu kegiatan akademik, seperti penerimaan mahasiswa baru, pembuatan kurikulum, pembuatan jadwal mata pelajaran, serta pengelolaan data guru dan siswa. Penelitian ini akan melakukan pengukuran kompleksitas terhadap *web service* sekolah dengan menggunakan metode *complexity class diagram*.

2. METODOLOGI PENELITIAN

2.1. Dasar Teori

2.1.1. *Web Service*

Web service adalah generasi baru dari aplikasi *Web*. Mereka mandiri, menggambarkan diri sendiri, aplikasi modular yang dapat dipublikasikan, ditemukan, dan dipanggil di seluruh *Web*. *Web service* menjalankan fungsi, yang bisa berupa apa saja dari permintaan sederhana hingga proses bisnis yang rumit. Setelah layanan *Web* digunakan, aplikasi lain (dan layanan *Web* lainnya) dapat menemukan dan menjalankan layanan yang digunakan.

Web service juga merupakan sebuah layanan yang diperuntukkan agar sebuah *platform* dapat berkomunikasi dengan *platform* yang lain. Sehingga sebuah sistem yang menggunakan *web service* dapat mendukung fitur *multi-platform*. Bentuk umum umum dari *web service* adalah WSDL (*Web Service Description Language*) dan REST (*Representational State Transfer*). WSDL menggunakan komunikasi SOAP (*Simple Object Access Protocol*) sedangkan REST menggunakan komunikasi dengan PUT, GET, POST dan memiliki *return* seperti HTML, XML, JSON, atau struktur yang ditentukan sendiri.

2.1.2. *Class Diagram*

Class diagram adalah deskripsi yang paling penting dan paling banyak digunakan dari sebuah sistem berbasis objek. *Class diagram* menunjukkan struktur statis dari class-class inti yang membangun sistem. *Class diagram* menampilkan *attribute* dan *method* pada setiap *class*. *Class diagram* juga menampilkan *relation* yang terdapat di antara setiap *class*.

Dalam *class diagram*, kelas-kelas tersebut diatur dalam kelompok-kelompok yang memiliki karakteristik yang sama. *Class diagram* menyerupai diagram alur di mana *class* digambarkan sebagai kotak, masing-masing kotak memiliki tiga persegi panjang di dalamnya. Kotak atas berisi nama *class*; kotak tengah berisi atribut *class*; persegi panjang yang lebih rendah berisi *method*, juga disebut *operation*, *class*. Garis, yang mungkin memiliki panah di salah satu atau kedua ujungnya, hubungkan kotak. Garis-garis ini mendefinisikan hubungan, juga disebut asosiasi, antara kelas-kelas.

2.1.3. Metrik Skala Kompleksitas *Class Diagram*

Metrik skala kompleksitas *class diagram* merepresentasikan tingkat kerumitan *class* yang digambarkan dengan *class diagram* pada suatu program. Berdasarkan *Object Oriented Programming* (OOP) yang merupakan bahasa pemrograman yang telah dirancang di sekitar *object* daripada "prosedur" dan informasi aktual daripada pemrograman logika. Pada dasarnya, sebuah program adalah serangkaian *method* yang dapat dieksekusi yang mengambil data sebagai *input*, melakukan pemrosesan, dan kemudian memberikan hasil yang diinginkan. Upaya aktual atau fondasi pemrograman difokuskan pada cara menulis logika daripada definisi data. OOP yang terutama menekankan bahwa *object* dalam paradigma pemrograman adalah *object* yang harus diubah sesuai dengan kebutuhan dan penggunaan kita dan bukan logika yang diperlukan di balik manipulasinya. *class* terdiri dari *method* (*Method*), atribut (*Attribute*), dan relasi

(*Relation*). Sehingga, metrik skala dan kompleksitas class diagram dapat diformulasikan sebagai berikut:

$$Class = (0.637 \cdot \sum Method) + (0.258 \cdot \sum Attribute) + (0.105 \cdot \sum Relation) \quad (1)$$

2.1.4. Metrik Skala Kompleksitas Method

Metrik skala kompleksitas *method* merepresentasikan tingkat kerumitan *method* pada suatu *class*. Kompleksitas *method* ditentukan skala kompleksitas alur program yang direpresentasikan menggunakan *flowchart*. Namun berdasarkan batasan masalah, perhitungan dilakukan dengan mengabaikan kompleksitas *method*, sehingga diasumsikan kompleksitas setiap *method* bernilai 1.

2.1.5. Metrik Skala Kompleksitas Attribute

Metrik skala kompleksitas attribute merepresentasikan tingkat kerumitan atribut-atribut pada suatu class. skala kompleksitas attribute ditentukan berdasarkan tipe data (DT) dan lebar datanya (DW). Setiap tipe data memiliki bobot tertentu berdasarkan tabel berikut.

Tabel 1. Skala Kompleksitas Attribute

Tipe Data	Bobot
Bit	0.01
Tinyint, year	0.06
Smallint	0.11
Char, binary, tinytext, tinyblob	0.12
Text, blob, varchar, varbinary	0.16
Mediumint, time	0.17
Int, float, timestamp	0.22
Mediumtext, mediumblob	0.24
Bigint, double, decimal, real, numeric, datetime, time	0.44
Longtext, longblob	0.48

Skala kompleksitas Attibut dapat diformulasikan sebagai berikut:

$$Attribute = DT \cdot DW \quad (2)$$

2.1.6. Metrik Skala Kompleksitas Relation

Metrik skala kompleksitas relation merepresentasikan tingkat kerumitan relasi antara class satu dengan yang lain pada suatu class diagram. Kompleksitas Relasi ($Relation$) dalam class diagram didapatkan dari penjumlahan berbagai tipe relasi yang ada dalam class diagram tersebut. Jenis-jenis relasi yang ada dalam class diagram adalah Inheritance ($Inheritance$), Association ($Association$), Composition ($Composition$), dan Aggregation ($Aggregation$). Skala kompleksitas relation dapat diformulasikan sebagai berikut:

$$Relation = (a \cdot Inheritance) + (b \cdot Association) + (c \cdot Composition) + (d \cdot Aggregation) \quad (3)$$

3. HASIL DAN PEMBAHASAN

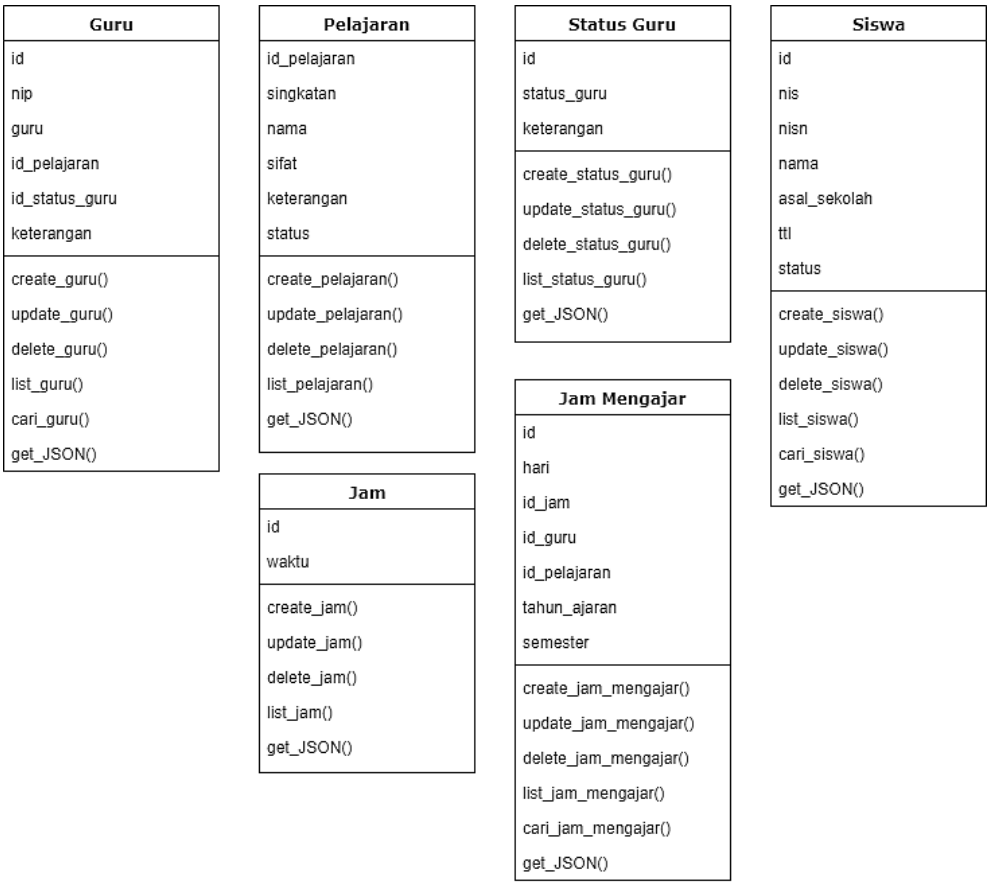
3.1. *Class Diagram*

Penelitian ini menggunakan 3 sistem kelas diagram, yakni: Akademik, Perpustakaan, dan Keuangan dengan menerapkan arsitektur Microservices dan berkomunikasi antara satu dengan yang lain dengan protokol REST. Dengan menggeneralisasi setiap tipe data memiliki bobot sebagai berikut:

String = 256
Int = 171
Enum = 100
DateTime = 7

3.1.1. Akademik

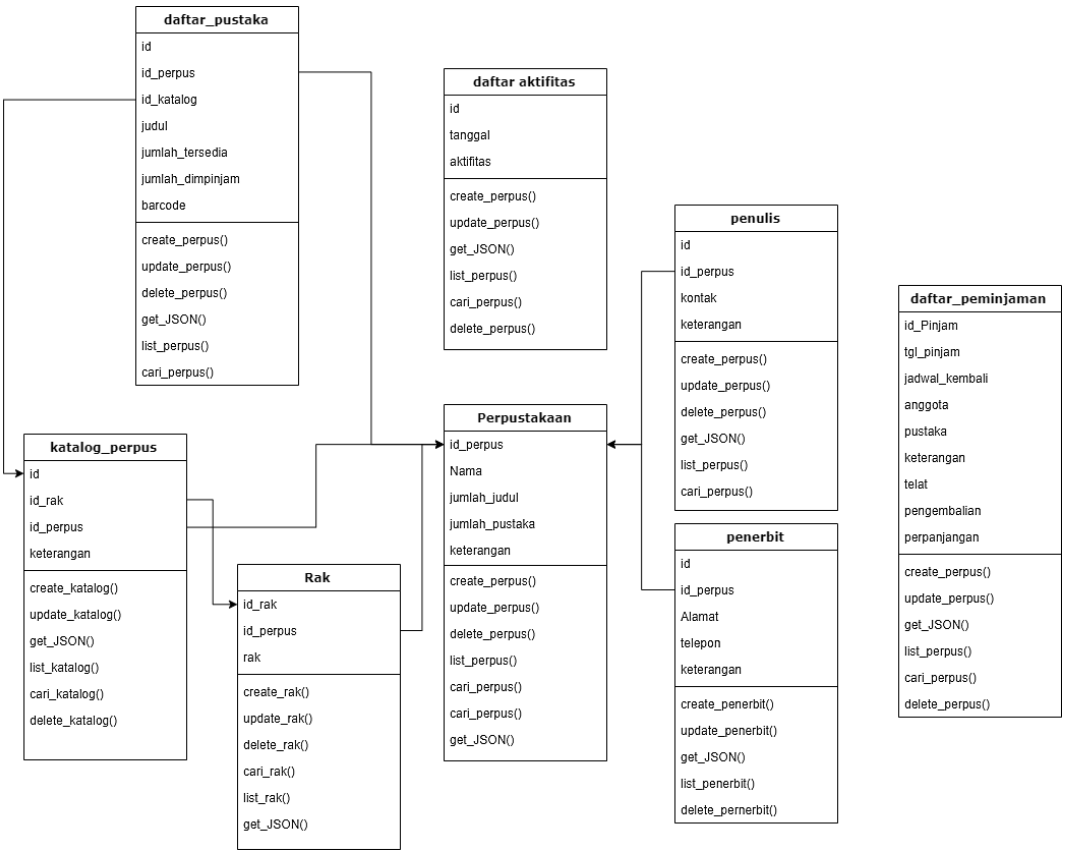
Class Diagram Akademik terdiri dari 6 kelas dengan total 31 *attribute* dan 34 *method*.



Gambar 1. Class Diagram Akademik

3.1.2. Perpustakaan

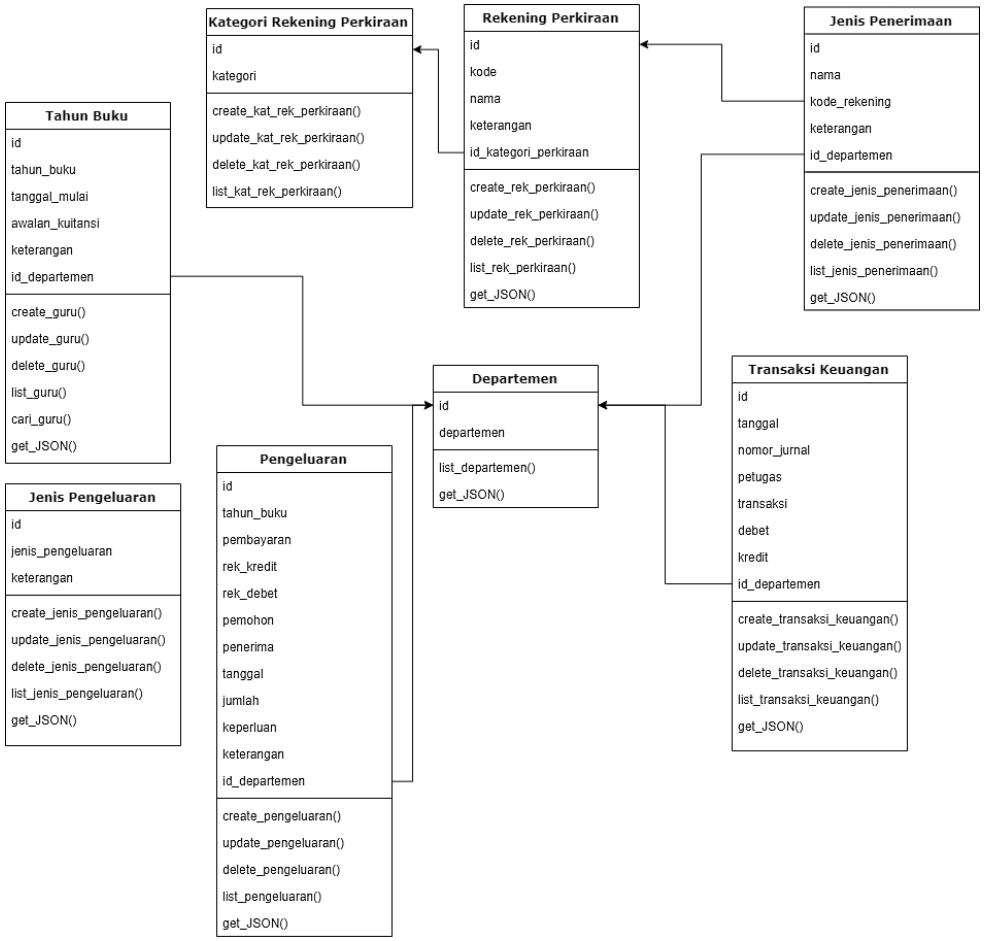
Class Diagram Akademik terdiri dari 8 kelas dengan total 31 *attribute* dan 34 *method*.



Gambar 2. Class Diagram Perpustakaan

3.1.3. Keuangan

Class Diagram Akademik terdiri dari 8 kelas dengan total 43 *attribute* dan 37 *method*.



Gambar 3. Class Diagram Keuangan

3.2. Skala Kompleksitas *Attribute*

Berikut persamaan dari skala kompleksitas *Attribute*.

$\sum Attribute = DT \cdot DW$
$Attribute\ Akademik = 1221.12$
$Attribute\ Perpustakaan = 611.94$
$Attribute\ Keuangan = 1509.42$
$\sum Attribute = 3342.49$

3.3. Skala Kompleksitas *Method*

Berikut persamaan dari skala kompleksitas *Method*.

$\Sigma Method$ = (*Method* Akademik)
+ (*Method* Perpustakaan)
+ (*Method* Keuangan)
= 118

3.4. Skala Kompleksitas *Relation*

Berikut persamaan dari skala kompleksitas *Relation*.

$\Sigma Relation$ = (*Relation* Akademik)

+ (*Relation* Perpustakaan)
+ (*Relation* Keuangan)
= 0 + 7 *Association*
+ 6 *Association*
= 13

3.5. kala Kompleksitas *Class Diagram*

Berikut persamaan dari skala kompleksitas *Class Diagram*.

$$Class = \left(0.258 \cdot \sum Attribute\right) + (0.637 \cdot \sum Method) + (0.105 \cdot \sum Relation)$$
$$= 862.36 + 75{,}17 + 1{,}37$$
$$= 938{,}9$$

4. SIMPULAN

Kompleksitas *Class Diagram* dapat dihitung dari penjumlahan kompleksitas *method* (*Method*), atribut (*Attribute*), dan relasi (*Relation*). Dari hasil perhitungan, diperoleh kesimpulan bahwa *Class Diagram* Sistem Informasi Akademik Sekolah diatas memiliki tingkat kompleksitas kode sebesar **993,32 unit.**

UCAPAN TERIMA KASIH

Pertama-tama, penulis menghaturkan terima kasih kepada dosen pengampu mata kuliah Manajemen Proyek, Bapak Ainul Yaqin, M.Kom atas bimbingan dalam penulisan serta arahannya. Lalu, terima kasih kepada teman-teman Andromeda yang telah memberikan dukungan moral serta bantuan kepada penulis. Penulis sangat mengapresiasi serta rasa syukur akan bantuan dari teman-teman Andromeda.

DAFTAR PUSTAKA

[1] Fensel, D., & Bussler, C. (2002). The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications, 1(2), 113–137. doi:10.1016/s1567-4223(02)00015-7
[2] Lanza, M., Marinescu, R., Ducasse, S. (2006). Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. *Springer Science & Business*, Berlin.
[3] Skogan, D., Grønmo, R., Solheim, I. (2004). Web Service Composition in UML. *IEEE International Enterprise Distributed Object Computing* Conference, Oslo .
[4] Souri, A., Mohammad ali Sharifloo, & Norouzi, M. (2011). Formalizing class diagram in UML. 2011 IEEE 2nd International Conference on Software Engineering and Service Science. doi:10.1109/icsess.2011.5982368
[5] Tan, R., & Caroline, W. A., (2018). *Pemrograman Web dengan PHP*. Google Play Books.
[6] Tonella, P., & Potrich, A. (2007). Reverse Engineering of Object Oriented Code. *Springer Science & Business,* New York.
[7] Urdhwareshe, A. (2016). Object-Oriented Programming and its Concepts. *Innovative Space of Scientific Research Journals*, Bridgeport.