PENGARUH PENGUJIAN STATIS TERHADAP KUALITAS PERANGKAT LUNAK SITAGIH PADA PT. SEMESTA MITRA SEJAHTERA (SMS)

Ahmad Faisol [1], Mira Orisa [2], Mochammad Ibrahim Ashari [3], Ni Putu Agustini [4]
[1,2] Teknik Informatika, Institut Teknologi Nasional Malang
[3,4] Teknik Elektro, Institut Teknologi Nasional Malang
mzfais@lecturer.itn.ac.id

ABSTRAK

Menghasilkan produk perangkat lunak yang memiliki kualitas tinggi, selalu menjadi tujuan akhir dari setiap pengembang perangkat lunak. Penjaminan kualitas tersebut dihasilkan dari proses yang terstruktur dan tidak mudah, oleh karena itu, proses pengembangan perangkat lunak tersebut perlu ditingkatkan atau diperbaiki untuk meningkatkan kualitas dari perangkat lunak yang dihasilkan. Salah satu tahapan untuk meningkatkan kualitas tersebut adalah dengan melakukan pengujian terhadap perangkat lunak yang dikembangkan. Banyak metode pengujian perangkat lunak yang dapat digunakan, salah satunya adalah pengujian statis. Pengujian ini dapat dilakukan tanpa harus melakukan eksekusi perangkat lunak dan tidak perlu menunggu proses pengembangan selesai dilakukan. Pada penelitian ini, dikembangkan aplikasi *Sitagih* yang akan diuji pada tiga kategori penerapan, yaitu berdasarkan modul perangkat lunak, komponen perangkat lunak, dan struktur perangkat lunak. Pengujian pada komponen perangkat lunak dilakukan dengan mengukur *Software Metric* menggunakan aplikasi *PhpMetrics*. Berdasarkan hasil pengujian diperoleh hasil bahwa masih terdapat beberapa *Class* yang belum lolos kasus uji. Selain itu, hasil pengukuran *metric* diperoleh tingkat *maintainability* yang rendah sehingga aplikasi *Sitagih* masih perlu perbaikan terutama di dokumentasi, penerapan algoritma yang lebih sederhana, dan penambahan sub kelas, khususnya pada *Class Order*. *Class tersebut* memiliki nilai *maintainability* di bawah 65 dan termasuk ke dalam kategori sulit untuk dirawat dan dikembangkan. Berdasarkan hasil penelitian ini, dapat dibuktikan bahwa pengujian statis memiliki pengaruh yang cukup signifikan terhadap kualitas perangkat lunak yang dihasilkan dari pengembangan.

Keyword : *pengujian aplikasi, kualitas perangkat lunak, software metrics*

1. PENDAHULUAN

Tujuan akhir dari proses rekayasa perangkat lunak adalah untuk menghasilkan perangkat lunak yang memiliki kualitas tinggi. Setiap perangkat lunak yang memiliki jaminan kualitas, dapat meminimalisir waktu untuk pengerjaan ulang, biaya yang lebih rendah, dan mempercepat waktu rilis. Salah satu komponen penjaminan kualitas pada proses rekayasa perangkat lunak adalah melakukan pengujian. Perangkat lunak yang gagal berfungsi karena minimnya aktivitas pengujian dapat mengakibatkan kualitas perangkat lunak yang rendah. Sehingga secara tidak langsung, aktivitas pengujian dapat menentukan kualitas dari perangkat lunak (Hendradjaya, 2017). Salah satu metode pengujian perangkat lunak adalah pengujian statis yang dapat dilakukan sejak lebih awal tanpa harus menunggu aplikasi selesai dikembangkan. Pengujian statis dapat diterapkan pada komponen atau unit program dan dokumen spesifikasi kebutuhan sistem.

Pengujian pada penelitian ini dilakukan pada aplikasi Sitagih. Aplikasi ini adalah sebuah program komputer berbasis web yang dikembangkan untuk mengelola data tagihan pengiriman barang pada PT. Semesta Mitra Sejahtera (SMS). Metode pengujian yang digunakan adalah pengujian statis terhadap komponen atau unit program dan dokumen desain aplikasi. Pengujian statis terhadap komponen dilakukan dengan cara melakukan analisis kode sumber (*code review)*, sedangkan pengujian statis pada dokumen desain dilakukan pada model dan struktur aplikasi.

2. TINJAUAN PUSTAKA

2.1. Penjaminan Mutu Perangkat Lunak

Penjaminan mutu perangkat lunak atau *Software Quality Assurance* merupakan rancangan serta ancangan sistematis terkait penilaian kualitas, standar produk, prosedur, dan proses pada perangkat lunak (Murugan dan Prakasam, 2013). Harapan dari jaminan perangkat lunak (SQA) adalah agar perangkat lunak yang dibangun bermutu tinggi.

Aktivitas penjaminan kualitas perangkat lunak memiliki tujuan untuk memantau proses rekayasa perangkat lunak agar menghasilkan perangkat lunak yang berkualitas tinggi. Standar pengujian yang ada saat ini memiliki kelebihan dan kekurangan masing-masing. Pemilihan standar pengujian dalam pengembangan perangkat lunak sangat penting bagi perusahaan sesuai dengan kebutuhan perangkat lunak dan kebutuhan perusahaan tersebut (Alshammri, 2013).

2.2. Pengujian Perangkat Lunak

Menurut standar ANSI/IEEE 1059, pengujian perangkat lunak adalah proses menganalisis suatu entitas perangkat lunak untuk mendeteksi adanya perbedaan antara kondisi yang ada dengan kondisi yang diinginkan serta mengevaluasi fitur-fitur dari entitas perangkat lunak. Pengujian tersebut merupakan proses inti dari jaminan kualitas perangkat lunak (Siagian, 2018).

Hendradjaya (2017) menyetakan bahwa, pengujian atau aktivitas pengujian dapat menentukan kualitas dari perangkat lunak. Perangkat lunak yang tidak pernah diuji, maka tidak bisa dijamin kualitasnya. Tetapi perlu diperhatikan bahwa aktivitas penjaminan kualitas perangkat lunak bukan hanya aktivitas pengujian. Dua aktivitas tersebut saling komplementer, karena salah satu penyebab sebuah perangkat lunak gagal berfungsi adalah minimnya pengujian terhadap perangkat lunak tersebut.

2.3. Pengujian Statis

Pengujian statis memungkinkan dilakukan pengujian perangkat lunak tanpa mengeksekusi program, sehingga aktivitas pengujian dilakukan dengan menggunakan kode program sumber atau dokumentasi lainnya (Hendradjaya, 2017).

Kelebihan dari pengujian statis pada aplikasi adalah kemungkinan jumlah cacat yang dapat ditemukan jauh lebih cepat sehingga biaya perbaikan bisa lebih murah jika dibandingkan dengan melakukan pengujian dinamis (Homés, 2013).

3. METODE PENELITIAN

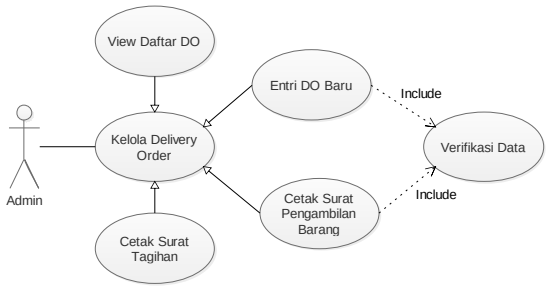
Metode penelitian yang digunakan adalah metode Pengujian Statis yang fokus pada tiga penerapan, yaitu:

1. Model aplikasi perangkat lunak
   Pengujian ini fokus pada proses analisis statis dokumen desain *Use Case Diagram* (Wibisono dan Baskoro, 2002).
2. Komponen aplikasi perangkat lunak.
   Pengujian ini fokus pada analisis kode sumber program yang digunakan untuk melihat apakah pemrogram sudah mengikuti standar pemrograman atau belum, seperti penamaan berkas dan variabel yang digunakan, penggunaan komentar pada setiap *method*, penamaan kelas dan objek, dan lain – lain.
3. Struktur aplikasi perangkat lunak
   Pengujian ini fokus untuk mendapatkan informasi tentang *class* dan *method* apa saja yang paling banyak digunakan, dan *method* yang memiliki kompleksitas yang tinggi. Perhitungan kompleksitas ini menggunakan bilangan siklomatik *(Cyclomatic Number / CN)* dengan rumus (Hendradjaya, 2017):
   CN = jumlah percabangan + 1 (1)

4. HASIL DAN PEMBAHASAN

4.1. Pengujian Berdasarkan Model Aplikasi Perangkat Lunak

Pengujian pada model aplikasi dilakukan dengan mempelajari desain *use case diagram* proses pencatatan *delivery order* dan menyiapkan kasus uji berdasarkan desain tersebut.



Gambar 1. Desain *Use Case Diagram* Proses Pencatatan *Delivery Order*

Desain *Use Case* pada Gambar 1 menunjukkan bahwa proses perekaman data pemesanan pengiriman dilakukan oleh seorang pengguna, yaitu admin. Setelah menerima nota pesanan dari *Supplier* produk pertanian, data akan diinputkan pada aplikasi. Hasil input data akan digunakan untuk mencetak surat pengambilan barang dan laporan tagihan bagi konsumen (peternak).

Berdasarkan hasil analisis kode sumber dari alur desain pada Gambar 1, maka dilakukan pengujian berdasarkan beberapa kasus uji sebagai berikut:

Tabel 1. Tabel Pengujian Berdasarkan Model Aplikasi Perangkat Lunak

No.	Kasus Uji	Harapan Hasil	Hasil Uji
1	Kasus input data yang benar	Sistem harus melakukan verifikasi dan menyimpan	Data tersimpan ke *database*
2	Kasus input data yang salah	Sistem harus melakukan verifikasi dan menolak data	Sistem memberikan pesan kesalahan data
3	Kasus input data angka yang melebihi batas minimal dan maksimal	Sistem harus menolak jika angka yang dimasukkan melebihi batas yang ditentukan	Sistem belum mampu membatasi input data angka dan tetap menerima data tersebut
4	Kasus input jika ada data yang sama	Sistem harus menolak dan menampilkan pesan kesalahan	Sistem menolak dan menampilkan kesalahan
5	Kasus menutup *form* dengan data yang belum tersimpan	Sistem harus menampilkan pesan	Sistem menampilkan pesan bahwa data belum disimpan
6	Kasus mencetak surat berdasarkan nomor order yang salah	Sistem harus menampilkan pesan kesalahan	Sistem menampilkan pesan bahwa nomor order tidak ditemukan
7	Kasus mencetak surat berdasarkan nomor order yang benar	Sistem harus menampilkan halaman cetak dan unduh	Sistem menampilkan halaman cetak dan unduh
8	Kasus mencetak surat tagihan tetapi	Sistem harus menampilkan	Sistem menampilkan

	saat belum ada transaksi	pesan belum ada transaksi	pesan belum ada transaksi
9	Kasus pengguna menekan tombol "bersihkan form" setelah ada *item order* yang belum disimpan	Sistem harus menampilkan pesan bahwa data belum disimpan	Sistem memberikan pesan peringatan bahwa data belum disimpan dan memberikan pilihan apakah akan disimpan atau tidak
10	Kasus pengguna menekan tombol "bersihkan form" sebelum ada data *item order* yang disimpan	Sistem harus membersihkan isian *form*	Sistem membersihkan isian *form* seperti kondisi awal

Berdasarkan Tabel 1, dapat disimpulkan bahwa dari 10 kasus pengujian yang dilakukan, terdapat 9 kasus yang sudah dinyatakan berhasil dan sesuai yang diharapkan, dan 1 kasus yang masih belum ditangani dengan benar. Hal ini menyebabkan masih ada kemungkinan kesalahan pengguna yang belum mampu ditangani oleh aplikasi.

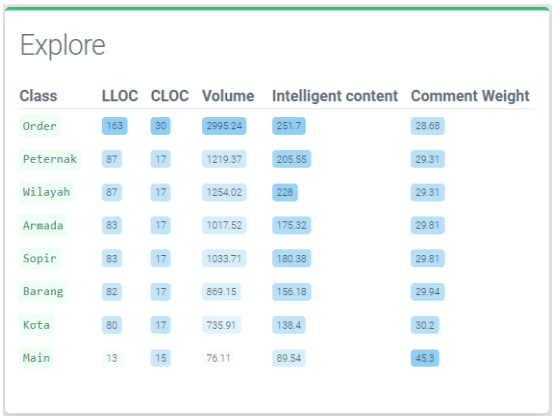
4.2. Pengujian Berdasarkan Komponen Aplikasi Perangkat Lunak

Pengujian pada komponen aplikasi dilakukan dengan menggunakan sebuah perangkat pengujian (*testing tools*) untuk mendapatkan nilai *metric* dari perangkat lunak. Perangkat yang digunakan adalah aplikasi *PhpMetrics* (Susanto, dkk., 2015). Aplikasi *PhpMetrics* ini dapat melakukan analisis dan review terhadap kode sumber program untuk menghasilkan nilai *complexity*, *maintainability*, *violations*, dan lain-lain.

4.2.1. *Lines Count* (LOC)

Lines Count berfungsi untuk menghitung banyaknya baris kode dalam suatu kode sumber program. Jumlah baris yang dihitung adalah semua baris kode termasuk komentar dan baris kosong (Rosenberg & Hyatt, dikutip dalam Susanto, dkk., 2015). Terdapat beberapa parameter yang diukur, yaitu LLOC (*lines count without empty line*), CLOC (lines count without multiline comments), *volume*, dan *Comment Weight*.

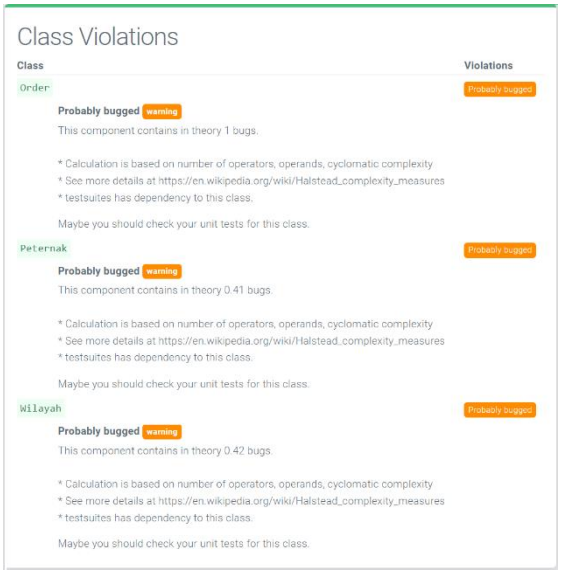
Berdasarkan hasil pengujian LOC yang ditunjukkan pada Gambar 2, diperoleh hasil bahwa dari 9 *Class* yang diuji, *Class Order* memiliki nilai yang paling besar. Selain itu, berdasarkan hasil parameter *Comment Weight*, dapat disimpulkan bahwa semua *Class* masih belum menerapkan dokumentasi dengan baik. Hal ini dapat berpengaruh pada proses pengembangan dan perawatan sistem, karena fungsi dari komentar pada baris kode adalah sebagai sumber dokumentasi dan kemudahan dalam memahami kode sumber program (Susanto, dkk., 2015).



Gambar 2. Hasil pengujian LOC menggunakan *PhpMetrics*

4.2.2. *Violations*

Violations berfungsi untuk menghitung kemungkinan banyaknya kemungkinan *error* dan *bugs* pada setiap *method* dalam *Class*.



Gambar 3. Hasil pengujian *Violations*

Hasil pengujian *Violations* dengan *PhpMetrics* pada aplikasi *Sitagih* diperoleh hasil yang ditunjukkan pada Gambar 3. Berdasarkan hasil pengujian tersebut tidak ditemukan adanya *error*, tetapi ditemukan 3 *Class* yang memiliki kemungkinan adanya *bugs* pada saat dioperasikan. *Bugs* tersebut ditemukan pada *Class Order*, *Peternak*, dan *Wilayah*, dengan nilai terbesar berada pada *Class Order* yaitu 1%. Hal tersebut disebabkan oleh beberapa faktor seperti adanya variabel yang dimungkinkan bernilai *null* atau kosong sehingga mengakibatkan kesalahan output bahkan *error* pada saat dioperasikan, atau karena nilai *Cyclomatic Complexity* yang besar dan berpengaruh pada performa aplikasi.

4.2.3. *Complexity & Defects*

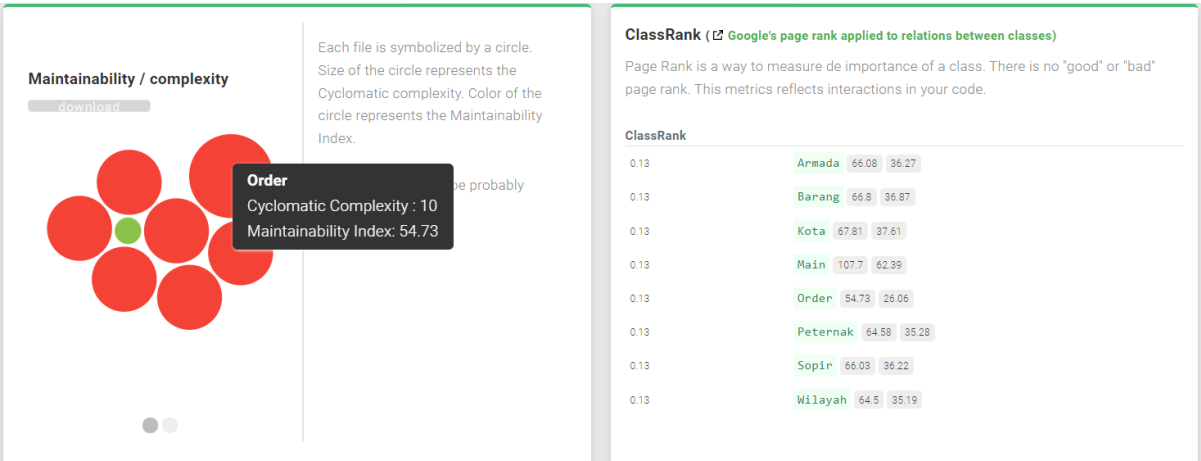
Parameter *Complexity & Defects* digunakan untuk menghitung nilai *Cyclomatic Complexity* (CC) dan kemungkinan adanya cacat pada aplikasi. Hasil pengujian pada aplikasi *Sitagih* ditunjukkan pada Gambar 4.

Class	WMC	Class cycl.	Max method cycl.	Relative data complexity	Relative structural complexity	Bugs	Defects	Relative system complexity
Armada	13	6	2	121.02	0.02	121	0.34	0.66
Barang	13	6	2	121.02	0.02	121	0.29	0.66
Kota	13	6	2	121.02	0.02	121	0.25	0.66
Main	1	1	1	1	0	1	0.03	0.15
Order	22	10	4	225.01	0.01	225	1	1.1
Peternak	13	6	2	169.02	0.02	169	0.41	0.66
Sopir	13	6	2	121.02	0.02	121	0.34	0.66
Wilayah	13	6	2	169.02	0.02	169	0.42	0.66

Gambar 4. Hasil pengujian *Complexity* menggunakan *PhpMetrics*

Berdasarkan Gambar 4, diperoleh hasil bahwa nilai WMC (*Weight Method per Class*) masih besar terutama pada *Class Order* dengan nilai 22. WMC dihitung berdasarkan jumlah kompleksitas *method* pada tiap *Class* yang berarti semakin tinggi nilai maka dapat berpengaruh pada seberapa banyak waktu dan usaha yang diperlukan untuk pengembangan dan perawatan suatu *Class* (Steidl, dkk., 2013).

Tingginya nilai WMC berakibat pada tingginya kemungkinan munculnya *bugs* pada *Class* yang ditunjukkan pada nilai 225, atau 1% dari jumlah *Relative Data Complexity*. Hal ini menunjukkan bahwa aplikasi *Sitagih* masih memerlukan banyak perbaikan karena nilai *Maintainability* yang kecil terutama pada *Class Order*, yang dapat dibuktikan pada Gambar 5.



Gambar 4. Hasil pengujian *Complexity* menggunakan *PhpMetrics*

Nilai *Maintainability* di bawah 65 berarti bahwa *Class* tersebut memiliki tingkat kesulitan yang tinggi dalam perawatan dan pengembangannya (Verisoft: Maintainability Index).

4.3. Pengujian Berdasarkan Struktur Aplikasi Perangkat Lunak

Pengujian berdasarkan struktur dilakukan untuk mengukur tingkat kompleksitas *method*, yang mana akan difokuskan pada *Class Order* karena dari hasil pengujian sebelumnya menunjukkan hasil yang kurang baik pada *Class* ini. Pada *Class Order* terdapat 14 *method* dan terdapat 13 *method* yang memiliki percabangan dan pengulangan.

Setelah menghitung jumlah kompleksitas, diperoleh hasil bahwa *method* yang paling tinggi nilainya adalah 4, yaitu pada *method ajax_add* (lihat Gambar 4). Berdasarkan nilai tersebut, harus dilakukan pengujian dengan jumlah minimal 4 kasus uji yang hasilnya dapat dilihat pada Tabel 2.

Tabel 2. Tabel Pengujian *Method ajax_add*

No.	Kasus Uji	Harapan Hasil	Hasil Uji
1	Kasus input data yang benar	Sistem harus melakukan verifikasi dan melanjutkan proses simpan	Sistem melakukan verifikasi dan mengembalikan nilai *TRUE*
2	Kasus input data yang salah	Sistem harus melakukan	Sistem memberikan

		verifikasi dan menolak data	pesan kesalahan data
3	Kasus input data baru dengan asumsi sudah melewati verifikasi data	Sistem harus menerima data input dan menyimpan ke *database*	Sistem menerima data dan menyimpan ke *database*
4	Kasus simpan *order* bernilai *TRUE*	Ubah variabel status menjadi *TRUE* dan melanjutkan untuk proses *item order*	Variabel status berubah menjadi *TRUE* dan lanjut ke proses *item order*
5	Kasus simpan *order* bernilai *FALSE*	Sistem harus menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan
4	Kasus simpan *item order* bernilai *TRUE*	Sistem harus menampilkan pesan bahwa proses penyimpanan berhasil	Sistem menampilkan pesan proses penyimpanan berhasil
5	Kasus simpan *item order* bernilai *FALSE*	Sistem harus menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan

Berdasarkan hasil pengujian pada Tabel 2, dapat disimpulkan bahwa semua kemungkinan kesalahan sudah mampu dikendalikan dengan benar oleh *method* tersebut. Akan tetapi, karena nilai kompleksitas yang tinggi, masih diperlukan perbaikan pada beberapa *method* di aplikasi Sitagih agar lebih mudah dalam pengembangan dan perawatan.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Pengujian statis pada aplikasi *Sitagih* telah dilakukan pada tiga kategori penerapan dan dapat disimpulkan bahwa nilai *Comment Weight* dan *Weight method per class* memiliki kriteria kurang bagus pada semua *Class* khususnya *Class Order*. Terdapat 3 *Class* yang memiliki kemungkinan tinggi munculnya suatu *bugs* pada saat dioperasikan karena masih terdapat beberapa variabel yang belum ditangani pada saat kondisi data *null*. Kriteria yang kurang bagus tersebut menyebabkan nilai Maintainability berada di bawah 65 yang mengakibatkan pengembangan dan perawatan sistem menjadi cukup sulit untuk dilakukan. Oleh karena itu, diperlukan perbaikan dan pengembangan struktur aplikasi dengan menambahkan beberapa dokumentasi / komentar, serta sub kelas dan *method* agar nilai *maintainability* menjadi lebih baik. Berdasarkan hasil pengujian tersebut, dapat dibuktikan bahwa pengujian statis sangat berpengaruh terhadap tingkat proses pengembangan aplikasi yang juga berdampak pada tingkat kualitas perangkat lunak. Semakin bagus hasil pengujian maka tujuan untuk menghasilkan perangkat lunak yang berkualitas tinggi dapat mudah tercapai, dan sebaliknya.

5.2. Saran

Pengujian statis yang sudah dilakukan hanya bagian kecil dari faktor penjaminan mutu perangkat lunak. Oleh karena itu, perlu dilakukan penelitian lebih lanjut terkait dengan metode pengujian aplikasi lainnya seperti *Black Box*, *Grey Box*, dan *White Box*.

DAFTAR PUSTAKA

[1] Alshammri, M. 2013. 'Problems in Software Quality Assurance and Reasons'. *IJCSI International Journal of Computer Science Issues*, Vol. 10, No. 1, hh. 325-327.

[2] Hendradjaya, B. 2017. *Konsep Dasar Pengujian Perangkat Lunak*. ITB Express, Bandung.

[3] Homés, B. 2013. Fundamentals of Software Testing, John Wiley & Sons, Inc.

[4] Murugan, C. S. dan Prakasam, S. 2013. 'A Literal Review of Software Quality Assurance'. *Januari Journal of Computer Applications*, Vol. 78, No. 8, hh. 25-30.

[5] Siagian, L. JM. 2018. *Otomatisasi Pengujian Perangkat Lunak (Software Test Automation)*. Deepublish, Yogyakarta.

[6] Steidl, D., dkk. 2013. *Quality Analysis of Source Code Comments*. CQSE GmbH, Garching B. Munchen, Germany.

[7] Susanto, M. I., dkk. 2015. 'Pengukuran Software Metric terhadap Implementasi Framework Laravel pada Pembangunan Aplikasi Berbasis Web. Studi kasus: Jurnal Logic'. *e-Proceeding of Engineering*, Vol. 2, No. 3, hh. 7731-7738.

[8] Verisoft. Measurement of Maintainability Index (MI) with Testwell CMT++ and CMTJava (Complexity Measurement Tools), https:// www.verifysoft.com/en_maintainability.html, on 13 Oktober 2021.

[9] Wibisono, W., dan Baskoro, F., 2002. 'Pengujian Perangkat Lunak dengan menggunakan Model Behaviour UML'. *Jurnal Ilmiah Teknologi Informasi, ITS*, Vol. 1, No 1, hh. 43-50.

IDF, SVD dan *Cosine Similarity* dapat membangun sebuah sistem temu kembali informasi pada undang-undang pemilu berdasarkan kasus pelanggaran pemilu.

2. Tingkat keberhasilan dari sistem temu balik informasi dengan metode LSI ini dapat dilihat melalui besarnya nilai uji kinerja sistem yang menghasilkan nilai *recall* sebesar 100%, *precision* sebesar 70 dan *f-measure* sebesar 82%. Pengujian oleh pengguna menghasikan persentase sebesar 89,21 % (sangat tinggi).

5.2. Saran

Berikut adalah saran yang peneliti berikan untuk penelitian kedepannya:

1. Pada penelitian ini masih menggunakan pasal dari UU no 10 tahun 2016 saja, untuk penelitian selanjutnya bisa ditambahkan undang-undang atau peraturan lain. Misal dari Perbub atau Perwali.
2. Karena jumlah pasal yang diproses banyak maka menambah lama waktu komputasinya. Pengembangan dan optimasi waktu komputasi yang cepat perlu dilakukan pada penelitian selanjutnya.

DAFTAR PUSTAKA

[1] Al-Anzi, F.S. dan AbuZeina, D. (2017). Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing. *Journal of King Saud University - Computer and Information Sciences*, 29 (2), 189–195. Tersedia pada https://doi.org/10.1016/j.jksuci.2016.04.001.

[2] Dwiyantoro, D. (2017). Sistem Temu Kembali Dengan Keyword (Deskriptif Menggunakan Recall Dan Precision Pada Judul, Subjek Opac Perpustakaan Universitas Gadjah Mada). *Khizanah al-Hikmah : Jurnal Ilmu Perpustakaan, Informasi, dan Kearsipan*, 5 (2), 164–175. Tersedia pada https://doi.org/10.24252/kah.v5i2a4.

[3] Froud, H., Lachkar, A. dan Ouatik, S.A. (2013). Arabic Text Summarization Based on Latent Semantic Analysis to Enhance Arabic Documents Clustering. *International Journal of Data Mining & Knowledge Management Process*, 3 (1), 79–95. Tersedia pada https://doi.org/10.5121/ijdkp.2013.3107.

[4] Hasan, M. ismail. (2018). Information Retrieval System artikel kesehatan menggunakan Pembobotan TF.IDF dan Latent Semantic indexing. *(Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim).*

[5] Khosrow-Pour, M. (2005). *Encyclopedia of Information Science and Technology (5 Volumes)*. Idea Group Reference.

[6] Wahib, A. dkk. (2015). Perangkingan Dokumen Berbahasa Arab Menggunakan Latent Semantic Indexing. *Jurnal Buana Informatika*, 6 (2), 83–92. Tersedia pada https://doi.org/10.24002/jbi.v6i2.411.

[7] Wahyuddin. (2017). Pemanfaatan sistem temu balik informasi di dinas perpustakaan dan kearsipan kabupaten barru skripsi. *(Doctoral dissertation, Universitas Islam Negeri alauddin makassar).*

[8] Yudho Baskoro, S., Ridok, A. dan Tanzil Furqon, M. (2015). Pencarian Pasal Pada Kitab Undang-Undang Hukum Pidana (Kuhp) Berdasarkan Kasus Menggunakan Metode Cosine Similarity Dan Latent Semantic Indexing (Lsi). *Journal of Enviromental Engineering and Sustainable Technology*, 2 (2), 83–88. Tersedia pada https://doi.org/10.21776/ub.jeest.2015.002.02.4.