Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 5, Mei 2019, hlm. 4787-4793 http://j-ptiik.ub.ac.id

Pembangunan Kakas Bantu Pengukuran *Maintainability* pada Tahap Perancangan Perangkat Lunak

Sigit Widodo[1], Fajar Pradana[2], Komang Candra Brata[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]sigitwido1@gmail.com, [2]fajar.p@ub.ac.id, [3]k.candra.brata@ub.ac.id

Abstrak

Maintainability didefinisikan sebagai kemudahan untuk memodifikasi sistem atau komponen dari perangkat lunak. Sebelumnya, terdapat penelitian dari Kumar dan Dhanda pada tahun 2015 yaitu "*Maintainability Measurement Model for Object Oriented Design*" yang menyatakan bahwa hasil nilai *maintainability* dapat diketahui berdasarkan *flexibility* dan *extendibility* pada *class diagram.* Penelitian tersebut menjelaskan perhitungan yang cukup memakan waktu lama karena harus melakukan proses perhitungan terhadap 4 metrik *Quality Model for Object-Oriented Design (*QMOOD). Oleh karena itu penelitian ini dimaksudkan untuk membangun sebuah kakas bantu pengukuran *maintainability* pada tahap perancangan perangkat lunak secara otomatis dengan mengimplementasikan perhitungan yang terdapat pada penelitian Kumar dan Dhanda tersebut. Pembangunan dilakukan dengan menggunakan metode *Waterfall*. Kakas bantu berbasis *desktop* dengan masukan berkas XML *class diagram* standar *simple format* dari aplikasi visual paradigm untuk mengetahui nilai *maintainability.* Kakas bantu tersebut memiliki *time based efficiency* 0,67 tugas/detik, dimana lebih cepat dibandingkan dengan pengukuran manual yang memiliki *time based efficiency* 0,0082 tugas/detik. Selain itu, memiliki *Overall relative efficiency* atau keberhasilan menyelesaikan tugas sebesar 100%.

Kata kunci: *maintainability*, XML *class diagram, metrik Quality Model for Object-Oriented Design (*QMOOD),

Abstract

Maintainability is defined as the ease of modifying a system or component of software. Previously there were studies from Kumar and Dhanda in 2015 namely "Maintainability Measurement Model for Object Oriented Design" which states that the results of maintainability values can be known based on flexibility and extendibility in the class diagram. The study explained that the calculation is quite time-consuming because it has to do the calculation process of the 4 metrics of Quality Model for Object-Oriented Design (QMOOD). Therefore this research is intended to build a tool for measuring maintainability at the stage of software design automatically by implementing the calculations found in the Kumar and Dhanda research. Development is done using the Waterfall method. The tools are desktop-based by entering the XML class diagram file in a standard simple format from the visual paradigm application to determine the value of maintainability. These tools have a time based efficiency of 0.67 tasks/second, which is faster than manual measurement which has a time based efficiency of 0.0082 tasks/second. Besides that, it has overall relative efficiency or the success of completing tasks by 100%.

Keywords*: maintainability, XML class diagram, Quality Model for Object-Oriented Design (QMOOD) metric*

1. PENDAHULUAN

Maintainability didefinsikan sebagai kemudahan untuk memodifikasi sistem atau komponen dari perangkat lunak (Izadkhah dan Hooshyar, 2017). Kemampuan *maintainability* sendiri merupakan atribut kualitas kunci yang menentukan keberhasilan suatu produk perangkat lunak (Kaur dan Pathak, 2014). Dalam hal pembiayaan sendiri, perawatan perangkat lunak adalah suatu bagian terbesar dari total biaya pembuatan aplikasi perangkat lunak. Beberapa studi memperkirakan bahwa *maintenance* membutuhkan hingga 80% dari total biaya yang digunakan (Ahn, Suh dan Kim, 2003). Sehingga perlu untuk mengetahui nilai

maintainability sejak awal pengembangan perangkat lunak. Sebelumnya terdapat penelitian tentang perhitungan kualitas perangkat lunak termasuk nilai *maintainability* dengan menggunakan kode program C++, yaitu "*A Source Code Quality Analysis Approach*" yang dilakukan oleh Tahira Iqbal pada tahun 2016. Sebagian besar studi mengukur *maintainability* atau menggunakan atribut yang berdampak pada *maintainability* dengan kode program. Meskipun pengukuran *maintainability* pada level kode program cukup akurat, tetapi hal tersebut mengakibatkan informasi tentang nilai *maintainability* terlambat untuk diketahui (Kumar dan Dhanda, 2015).

Kemudian terdapat penelitian lain yang berjudul "*Maintainability Measurement Model for Object Oriented Design*" menyatakan bahwa hasil nilai *maintainability* dapat diketahui berdasarkan *flexibility* dan *extendibility* pada *class diagram.* Dalam perhitungan yang dilakukan pada penelitian tersebut didapatkan tingkat korelasi sebesar 99% yang artinya sangat handal dan signifikan. Penelitian tersebut menjelaskan perhitungan yang cukup memakan waktu lama karena harus melakukan proses perhitungan terhadap 4 metrik QMOOD (*Quality Model for Object-Oriented Design)* yaitu CAM (*Cohesion Among Methods of Class*), DCC (*Direct Class Coupling*), NOP (*Number of polymorphic Method*) dan MFA (*Measure of Functional Abstraction*) untuk mendapatkan nilai *flexibility* dan *extendibility* kemudian barulah dilakukan perhitungan untuk mengukur nilai dari *maintainability* itu sendiri (Kumar dan Dhandra, 2015). Sehingga diperlukan aplikasi yang dapat mengukur nilai *maintainability* tersebut tanpa memakan waktu yang lama. Sebelumnya terdapat aplikasi untuk mengukur *maintainability* yaitu CAST *Software Maintainability.* Pada aplikasi tersebut *maintainability* dapat dievaluasi melalui survei kemudian secara manual memasukkan pertanyaan survei ke dalam aplikasi seperti berapa banyak FTE (*Full Time Equivalent*) yang diperlukan untuk proyek. Tentunya hal tersebut masih memerlukan usaha serta waktu yang tidak sedikit dalam mengukur *maintainability*.

Berdasarkan latar belakang masalah yang telah dijelaskan maka dilakukan penelitian untuk pembangunan kakas bantu yang sanggup untuk mengukur nilai dari *maintainability.* Pengukuran dilakukan pada tahap perancangan secara otomatis agar waktu yang dibutuhkan untuk menghitung nilai *maintainability* dapat dilakukan dengan lebih cepat serta tepat. Oleh karena itu digumakan perhitungan *maintainability* yang bersumber pada penelitian "*Maintainability Measurement Model for Object Oriented Design*" dengan tingkat korelasi sebesar 99%. Dengan adanya kakas bantu ini, maka saat merancang *class diagram* diharapkan hasil perancangan telah dapat memperkirakan nilai *maintainability*. Hal itu dapat mengakibatkan suatu kekurangan yang mempengaruhi kualitas perangkat lunak dapat di deteksi dari awal dengan cepat dan tepat.

2. DASAR TEORI

2.1. *Object Oriented Design (OOD) Metric*

Metric merupakan kuantitas yang terukur, kuantitas yang dimaksud pada penelitian ini yaitu pada *Object-Oriented Design* (OOD). OOD adalah proses dimana desainer mensintesis semua persyaratan yang dikumpulkan selama tahap analisis dan memetakannya kepada objek serta hubungan antar objek tersebut (Din dan Indris, 2009). OOD metric yang digunakan pada penelitian ini untuk menentukan nilai *maintainability* adalah CAM (*Cohesion Among Methods of Class*), DCC (*Direct Class Coupling*), MFA (*Measure of Functional Abstraction*) dan NOP (*Number of polymorphic Method*) pada jurnal penelitian *A Hierarchical Model for Object Oriented Design Quality Assessment* yang ditulis oleh Bansiya (Kumar dan Dhanda, 2015).

2.1.1. *Direct Class Coupling* (DCC)

Pada pengukuran DCC menunjukkan properti interdependensi di antara objek dalam desain. Pengukuran yang dilakukan menggunakan total jumlah dari kelas lain yang terkait langsung dengan kelas. Menggunakan nilai rata-rata DCC yaitu hasil DCC seluruh kelas dibagi jumlah kelas yang tersedia bila diterapkan pada desain secara keseluruhan (Fujita dan Pisanelli, 2007). Berikut adalah DCC Metric yang dirumuskan pada Persamaan 1.

$$\text{DCC}\,(class\ diagram) = \frac{\sum dcc}{\sum c} \quad (1)$$

Keterangan:
dcc = Jumlah kelas berbeda yang berelasi langsung dengan kelas (untuk masing-masing kelas)
c = Kelas yang ada pada *class diagram*

2.1.2 *Cohesion Among Methods of Class* (CAM)

Pada pengukuran metrik CAM menunjukan keterkaitan antara metode kelas berdasarkan daftar parameter atau metode. Perhitungan yang dilakukan menggunakan penjumlahan perpotongan parameter *method* dengan set independen maksimum dari semua tipe parameter di kelas (Fujita dan Pisanelli, 2007). Diasumsikan bahwa metode kelas, memiliki akses ke yang serupa dengan jenis parameter dan proses informasi yang terkait erat (Kaur dan Signh). Berikut adalah CAM Metric yang dirumuskan pada Persamaan 2 :

$$\text{CAM} = \frac{\sum \frac{\text{p}}{\text{m} \times \text{td}}}{\sum cs} \quad (2)$$

Keterangan:
p = Jumlah parameter berbeda dari setiap *method*
m = Jumlah *method*
td = Jumlah tipe data berbeda pada *class*
cs = Kelas yang mempunyai minimal 1 *method*

2.1.3 *Measure of Functional Abstraction* (MFA)

Pada pengukuran metrik MFA diartikan sebagai rata-rata dari semua kelas dalam suatu desain (dengan setidaknya satu metode tersedia) dari total jumlah metode yang diwarisi dibagi dengan jumlah total metode yang tersedia untuk kelas tersebut, berupa metode yang diwarisi ditambah dengan metode yang didefinisikan pada kelas(Fujita dan Pisanelli, 2007). Berikut adalah MFA Metrik yang dirumuskan pada Persamaan 3 :

$$\text{MFA} = \frac{\sum \frac{\text{mp}}{\text{ma}}}{\sum cs} \quad (3)$$

Keterangan:
mp = Jumlah method yang diwarisi
ma = Jumlah method yang tersedia
cs = kelas yang mempunyai minimal 1 method

2.1.4 *Number of Polymorphic Method* (NOP)

Metrik ini menunjukan berapa banyak *method* yang dapat menunjukkan perilaku polimorfik (Bansiya, 2002). Berikut adalah NOP Metric yang dirumuskan pada persamaan 4 :

$$\text{NOP} = \sum pl \quad (4)$$

Keterangan:
mp = Method yang memiliki sifat polimorfik

2.2 *Maintainability* Perangkat Lunak

Maintainability adalah ukuran dari seberapa mudah suatu sistem perangkat lunak dapat dipertahankan atau dilakukan pemeliharaan (*maintenance*). *Maintenance* dari perangkat lunak terdapat tiga jenis antara lain sebagai berikut (Sommerville, 2011):

1. *Maintenance* untuk memperbaiki kesalahan yang terjadi atau biasa disebut dengan *fault repair*.
2. *Maintenance* apabila terdapat perubahan seperti sistem operasi, perangkat keras, dan juga perangkat lunak pendukung lain pada lingkungan perangkat lunak yang disebut dengan *platform adaptation*.
3. *Maintenance* apabila terjadi penambahan fitur atau perubahan terhadap kebutuhan yang merupakan perubahan dari bisnis atau organisasi, hal tersebut biasa disebut dengan *functionality addition*/*system enhancement*.

Maintainability adalah faktor kualitas yang paling penting untuk memberikan perangkat lunak berkualitas baik. Kurangnya *maintainability* mengakibatkan pekerjaan yang berat pada saat sistem mengalami permasalahan ataupun penambahan fungsi baru (Kumar dan Dhanda, 2015). Terdapat 2 faktor yang mempengaruhi *maintainability* dari segi *object oriented design* yaitu *Extendibility* dan *Flexibility* (Bansiya, 2002). Untuk melakukan perhitungan terhadap nilai *maintainability* maka dengan *Statistical Package for the Social Sciences* (SPSS), nilai dari koefisien dihitung dan model *maintainability* dirumuskan seperti Persamaan 5 berikut (Kumar dan Dhanda, 2015).

$$M = 4.749 - (0.398 \times Flexibility) + (0.023 \times Extendibility) \quad (5)$$

Keterangan:
M = *Maintainability*

2.2.1. *Flexibility*

Flexibility berkaitan dengan kemampuan ataupun upaya yang dilakukan untuk menjaga suatu kegiatan pemeliharaan adaptif atau menambah/memodifikasi/menghapus fungsi tanpa merusak sistem saat ini. Contoh pada *flexibility* yaitu pada *Teacher Support Software* (TSS), sistem tersebut dapat digunakan oleh

semua level pengguna yang artinya dapat disesuaikan ketika diterapkan untuk SD, SMP ataupun SMA (Galin, 2018). Dalam perhitungan untuk mendapatkan nilai *flexibility* yang mendukung untuk memperoleh *maintainability* diperlukan perhitungan dengan menggunakan nilai *coupling* diukur dengan metrik DCC, *cohesion* diukur dengan metrik CAM dan *inheritance* diukur dengan metrik MFA (Kumar dan Dhanda, 2015). Dengan *Statistical Package for the Social Sciences* (SPSS), nilai dari koefisien dihitung dan model *flexibility* dirumuskan seperti persamaan 6 berikut.

$$F = 3.878 + (0.153 \times \text{Coupling}) + (-2.664 \times \text{Cohesion}) + (9.211 \times \text{Inheritance}) \quad (6)$$

Keterangan:
F = *Flexibility*

2.2.2. *Extendibility*

Extendibility berkaitan dengan kemudahan mengadaptasi perangkat lunak terhadap perubahan spesifikasi. Contoh pada *extendibility* suatu maskapai penerbangan yang memiliki sistem reservasi tiket tidak hanya membayar untuk memiliki sistem tersebut tetapi juga untuk biaya pengembangan jika terdapat spesifikasi kebutuhan yang baru dibutuhkan (Mayer, 1997). Dalam perhitungan untuk mendapatkan nilai *extendibility* yang mendukung untuk memperoleh *maintainability* diperlukan perhitungan dengan menggunakan nilai *coupling* diukur dengan metrik DCC, *cohesion* diukur dengan metric CAM dan *polymorphism* diukur dengan metrik NOP (Kumar dan Dhanda, 2015). Dengan *Statistical Package for the Social Sciences* (SPSS), nilai dari koefisien dihitung dan model *extendibility* dirumuskan seperti Persamaan 7 berikut :

$$E = 9.859 + (-11.186 \times \text{Coupling}) + (1.101 \times \text{Cohesion}) + (3.102 \times \text{Polymorphism}) \quad (7)$$

Keterangan:
E = *Extendibility*

2.3 Perhitungan Efisiensi

Jika dilihat pada ISO-9241, efisiensi dapat diartikan seberapa besar usaha atau tenaga yang dibutuhkan untuk memastikan suatu tujuan atau tugas dapat berjalan dengan sukses. Efisiensi sendiri termasuk kedalam kebutuhan produk yang merukapan suatu kebutuhan non-fungsional (Sommerville, 2011). Terdapat dua jenis cara dalam melakukan perhitungan efisiensi antara lain adalah *time based efficiency* (efisiensi waktu) serta *overall relative efficiency* (efisiensi secara keseluruhan) (Sergeev, 2010). *Time based efficiency* diperoleh dengan mengamati pengguna sistem ketika menyelesaikan tugas dengan waktu yang pengguna perlukan. Sedangkan *overall relative efficiency* didapatkan dari waktu presentase dalam pengguna butuhkan untuk keberhasilan menyelesaikan tugas yang berkaitan dengan waktu total yang semua pengguna butuhkan. Berikut adalah persamaan perhitungan *time based efficiency* pada persamaan 8 dan persamaan untuk perhitungan *overall relative efficiency* pada persamaan 9.

$$TBE = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR} \quad (8)$$

$$ORE = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} * 100\% \quad (9)$$

Keterangan:
TBE : *Time Based Efficiency*
ORE : *Overall Relative Efficiency*
N : Total dari jumlah tugas yang dikerjakan
R : Total dari jumlah pengguna
n_{ij} : hasil tugas i yang sukses dilakukan oleh pengguna j (berhasil = 1, gagal = 0)
t_{ij} : Waktu pengguna untuk melakukan tugas

3. METODOLOGI PENELITIAN

Dalam metodologi penelitian membahas metode dan tahap dalam melakukan penelitian. Pada Gambar 1 adalah tahapan penelitian, dimana studi literatur adalah tahapan pertama, kemudian tahap analis kebutuhan, tahap perancangan dan implementasi sistem, setelah itu pengujian sistem dan terakhir adalah penutup.

Pada tahap studi literatur mempelajari mengenai pustaka ataupun referensi yang berkaitan terhadap permasalahan penelitian ini yaitu pengembangan perangkat lunak, rekayasa perangkat lunak, *object oriented design metric*, *maintainability*, *flexibility*, *extendibility* perangkat lunak, pengujian perangkat lunak dan juga mempelajari tentang teknologi pengembangan yaitu jaxp, swing serta sqlite.

Pada tahap analisis kebutuhan berisi tentang deskripsi umum sistem, kemudian identifikasi aktor, identifikasi serta spesifikasi kebutuhan fungsional dan non-fungsional, serta pemodelan kebutuhan berupa *use case diagram*. Selain itu

tahap analisis juga berisi contoh perhitungan manual untuk mengukur nilai *maintainability*.

Gambar 1 Tahapan Penelitian

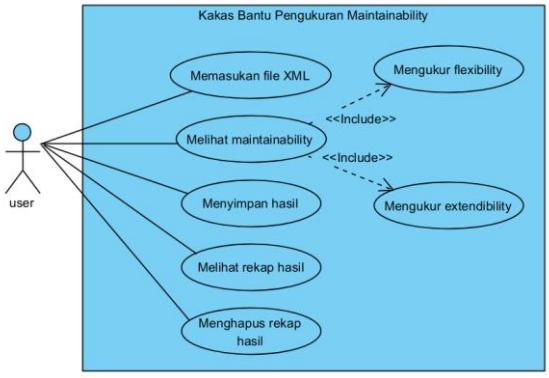
Pada tahap perancangan sistem, dilakukan perancangan arsitektur, algoritme, basis data serta antarmuka. Selanjutnya juga dilakukan pembahasan terhadap hasil dari implementasi pada penelitian ini yang sesuai dengan perancangan yang telah dilakukan.

Kemudian pada tahapan pengujian bertujuan untuk dapat memastikan bahwa sistem mampu berjalan seperti apa yang terdapat pada kebutuhan. Dilakukan pengujian unit dengan metode *whitebox*, pengujian integrasi yang menggunakan pendekatan *top down*, dan pengujian validasi dengan metode *blackbox*.

Selanjutnya, tahapan terakhir adalah penutup berupa pengambilan kesimpulan serta saran yang dapat diperoleh berdasarkan penelitian Pembangunan Kakas Bantu Pengukuran *Maintainability* pada Tahap Perancangan Perangkat Lunak.

4. ANALISIS KEBUTUHAN

Pada penelitian ini, analisis kebutuhan dimulai pada pembahasan mengenai deskripsi umum, kemudian identifikasi pengguna dengan hasil identifikasi pengguna sistem memiliki 1 aktor yaitu *user*. Setelah itu menentukan kebutuhan fungsional beserta spesifikasinya dimana sistem memiliki 7 kebutuhan fungsional. Setelah itu menentukan kebutuhan non-fungsional dimana sistem memiliki 1 kebutuhan non-fungsional.



Gambar 2 *Use Case Diagram* Sistem

Selanjutnya melakukan pemodelan kebutuhan yang berupa *usecase diagram* yang terdapat pada Gambar 1 dan *usecase scenario* untuk menjelaskan lebih rinci tentang pemodelan fungsional.Selain itu juga terdapat perhitungan manual untuk mengukur *maintainability* menggunakan *class diagram* dari *Sales Order System.*

5. PERANCANGAN & IMPLEMENTASI

Pada perancangan dibuat rancangan sesuai dengan hasil dari analisis kebutuhan dimana memuat tentang perancangan arsitektur antara lain *sequence diagram* serta *class diagram*, kemudian perancangan basis data, perancangan algoritme yang menghasilkan pseudocode dari fungsional sistem, serta perancangan antarmuka sistem.

Pada tahap implementasi adalah proses lanjutan setelah dihasilkan rancangan sistem. Implementasi sistem yang dilakukan terdapat penjelasan mengenai spesifikasi sistem yang digunakan dalam proses implementasi, kemudian kode sumber serta implementasi yang disajikan dalam bentuk antarmuka. Implementasi yang dibuat telah sesuai dengan hasil rancangan yang sebelumnya telah diperoleh.

6. PENGUJIAN DAN PEMBAHASAN

Pada penelitian ini pengujian sistem bertujuan untuk dapat mengetahui bahwa sistem telah mampu berjalan sesuai spesifikasi kebutuhan yang ada. Pengujian yang telah dilakukan untuk menguji sistem pengukuran nilai *maintainability* pada tahap perancangan perangkat lunak yaitu pengujian unit, kemudian pengujian integrasi serta pengujian validasi. Untuk pengujian unit menggunakan metode

white box dengan mendapatkan hasil status valid pada seluruh kasus uji. Kemudian pada pengujian integrasi menggunakan pendeketan *top down* juga mendapatkan status valid. Setelah itu untuk pengujian validasi menggunakan seluruh kebutuhan fungsional mendapatkan hasil status valid.

Kemudian pengujian validasi dilakukan juga untuk menguji kebutuhan non-fungsional dengan menghitung efesiensi dari sistem. Pada pengujian ini dilakukan oleh 1 orang yang menguji dan menggunakan 3 berkas uji yang mempunyai jumlah kelas yang berbeda. Berdasarkan hasil pengujian efesiensi menunjukan bahwa hasil efesiensi pengukuran sistem memiliki *time based efficiency* 0,67 tugas/detik, dimana lebih cepat dibandingkan dengan pengukuran manual yang memiliki *time based efficiency* 0,0082 tugas/detik. Kemudian memiliki efisiensi secara keseluruhan dalam keberhasilan menyelesaikan tugas (*Overall relative efficiency)* sebesar 100%. Dengan demikian sistem telah mampu memenuhi kebutuhan non-fungsional yaitu mempunyai efisiensi 100% dalam keberhasilan melakukan pengukuran dan lebih cepat jika dibandingkan dengan pengukuran manual.

6.1 Pembahasan Hasil

Dalam pembangunan kakas bantu pengukuran *maintainability* pada tahap perangkat lunak juga diuji dengan masukan *class diagram* proyek perangkat lunak *open source* yaitu *Photo Lab Management System* dan *Bug Tracking*. Selain itu juga diuji dengan *class diagram* proyek *web service* API Perpustakaan Universitas Brawijaya serta *class diagram Sales Order System System*. Sesuai dengan pengukuran yang dilakukan maka nilai *Maintainability* terbaik pada tiga kasus uji tersebut adalah *web service* API Perpustakaan Universitas Brawijaya dengan nilai 3,706, kemudian *Photo Lab Management System* dengan nilai 2,738, setelah itu *Sales Order System System* dengan nilai 2,604 dan *Bug Tracking* dengan nilai 1,494.

7. KESIMPULAN

Berdasarkan hasil penelitian pada pembangunan kakas bantu pengukuran *maintainability* pada tahap perancangan perangkat lunak didapatkan kesimpulan berikut ini:

1. Pembangunan kakas bantu pengukuran *maintainability* pada tahap perancangan perangkat lunak mempunyai 7 kebutuhan fungsional serta 1 kebutuhan non-fungsional. Untuk kebutuhan fungsional diperoleh dari jurnal utama oleh Rajendra Kumar dan Namrata Dhanda dengan judul *Maintainability Measurement Model for Object-Oriented Design* untuk dasar penelitian serta dilakukan pengembangan. Pengembangan yang dilakukan menggunakan *waterfall model* karena seluruh kebutuhan sudah jelas didapatkan pada tahap awal. Kemudian dalam perancangan sistem diperoleh perancangan komponen, perancangan arsitektur, perancangan basis data serta perancangan antarmuka. Setelah itu berdasar kepada implementasi diperoleh hasil implementasi yang merujuk terhadap tahap perancangan yaitu implementasi basis data, implementasi kode program, serta implementasi antar muka sistem. Kemudian pengujian dilakukan pengujian unit dengan metode *whitebox*, pengujian integrasi dengan pendekatan *top-down testing* serta pengujian validasi dengan metode *blackbox*. Berdasarkan pengujian yang telah dilakukan, diperoleh hasil valid terhadap semua kasus uji, dengan demikian kkas bantu yang pada penelitian ini layak untuk digunakan.
2. Hasil efesiensi pengukuran sistem memiliki *time based efficiency* 0,67 tugas/detik, dimana lebih cepat dibandingkan dengan pengukuran manual yang memiliki *time based efficiency* 0,0082 tugas/detik. Kemudian memiliki efisiensi secara keseluruhan dalam keberhasilan menyelesaikan tugas (*Overall relative efficiency)* sebesar 100%.

8. DAFTAR PUSTAKA

Ahn, Yunsik, Jungseok Suh, Seungryeol Kim, and Hyunsoo Kim. 2003. The Software Maintenance Project Effort Estimation Model Based on Function Points. *Journal of Software Maintenance and Evolution* 15 (2): 71–85.

Bansiya, Jagdish. 2002. A Hierarchical Model for Object Oriented Design Quality Assessment. *IEEE Transaction of Software Engineering* 28 (1): 4-17

Fujita, H. & Pisanelli, D. M., 2007. *New Trends in Software Methodologies, Tools and Techniques.* 161 penyunt. Netherlands: IOS Press.

Galin , Daniel. 2018. Software Quality Factors (Attributes). *Software Quality: Concepts and Practice* , IEEE.

Kaur, K & Signh, H. (2009) Exploring Design Level Class Cohesion Metrics. Departement of Computer Science and Engineering.

Kumar, Rajendra, and Namrata Dhanda. 2015. Maintainability Measurement Model for Object- Oriented Design. 4 (5): 68–71.

Meyer, Bertrand. 1997. Object-Oriented Software Construction Second Edition. US America : Interactive Software Engineering, Inc.

Sergeev, A., 2010. *Efficiency*. [Online] Available at: http://ui-designer.net/usability/efficiency.htm [Diakses 12 2 2019].

Sommervile, Ian. 2011. *Software Engineering 9th Edition.* US America: Pearson Education, Inc.