Pengujian Nonfungsional dengan Pendekatan *McCall's Factor* pada Perspektif *Product Revision*

Sheilla Fajriah Muthmainnah
Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
19523140@students.uii.ac.id

Hanson Prihantoro Putro, S.T., M.T.
Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
hanson@uii.ac.id

Abstract*—Perangkat lunak merupakan kumpulan program yang berisikan perintah atau prosedur yang digunakan untuk mengolah informasi. Dalam pengembangannya, banyak ditemukan permasalahan pada suatu produk perangkat lunak. Misalnya seperti banyak terdapat *bug* dalam pengembangan sebuah fitur. Oleh karena itu, diperlukan sebuah pengujian untuk melihat perangkat lunak dapat berfungsi dengan baik atau tidak. Salah satu metode yang digunakan dalam pengujian perangkat lunak secara manual adalah menggunakan metode *McCall's Factor*. Penelitian ini bertujuan untuk menguji perangkat lunak dengan menggunakan pendekatan *McCall's Factor* pada aspek *product revision* terhadap aplikasi Ivent. Proses pengujian pada penelitian ini memiliki enam tahapan, yaitu mengidentifikasi jenis tes yang dilakukan, lalu menentukan metrik pengujian dan perumusan pada setiap karakteristik yang telah ditentukan. Kemudian melakukan pengaturan lingkungan uji untuk menentukan syarat *software* telah benar-benar siap untuk diuji dan selanjutnya dilakukan pengujian. Setelah pengujian dilakukan tahap selanjutnya akan dilakukan penutupan siklus uji. Hasil penelitian ini menunjukkan perangkat lunak berbasis web Ivent masih sangat tidak baik, jika dilihat dari hasil perhitungan faktor nonfungsional dengan nilai *Maintainability* 13%, *Flexibility* 8.75%, dan *Testability* 11%. Penelitian ini menjadi acuan untuk perbaikan dan pengembangan perangkat lunak oleh tim *developer* agar lebih baik.

Keywords—Software quality, McCall's Factor, Product Revision, Ivent.

I. PENDAHULUAN

Perangkat lunak merupakan kumpulan program yang berisikan perintah atau prosedur yang digunakan untuk mengolah informasi. Salah satu faktor penting untuk memastikan bahwa sebuah perangkat lunak bekerja sesuai dengan fungsinya, maka diperlukan proses pengujian perangkat lunak [1]. Proses pengujian perangkat lunak bertujuan untuk mendeteksi keberadaan *bug* dalam sebuah perangkat lunak yang tidak diharapkan dalam pengembangan sistem [2]. Dalam kehidupan sehari-hari banyak ditemukan permasalahan pada suatu produk perangkat lunak. Misalnya banyak terdapat *bug* dalam pengembangan sebuah fitur baru yang ada pada sistem. Oleh karena itu diperlukan sebuah pengujian nonfungsional.

Pengujian nonfungsional harus dilakukan karena pengujian nonfungsional menguji aspek-aspek seperti kinerja, kegunaan, keandalan, keamanan, muatan, dan kompabilitas dari aplikasi perangkat lunak agar dapat berjalan dengan baik dalam kondisi yang berbeda. Pengujian nonfungsional memiliki berbagai metode pengujian, salah satu metode pengujiannya adalah *McCall's Factor*. Pengujian dengan metode *McCall's Factor* dipilih karena menggunakan skala dan bobot untuk memberikan penilaian kualitatif terhadap faktor-faktor yang dinilai, hal ini dapat membantu mengidentifikasi area yang memerlukan perbaikan atau peningkatan.

Metode *McCall's Factor* adalah model pengujian tertua yang dikembangkan pada tahun 1996 yang dikembangkan oleh McCall Richards dan Walter pada tahun 1977. *McCall* mengklasifikasikan tiga hal penting yang dapat memengaruhi kualitas perangkat lunak yang dikenal sebagai *product perspective* [3]. Menurut *McCall* tiga hal penting tersebut adalah transisi produk, operational produk, dan revisi produk. Pengujian tahap revisi produk dipilih karena membantu dalam memastikan bahwa perangkat lunak yang direvisi memenuhi standar kualitas yang diharapkan. Hal ini membantu dalam mengidentifikasi dan memperbaiki cacat atau masalah yang mungkin muncul setelah melakukan perubahan pada produk. Apabila revisi produk tidak diperhatikan dapat menyebabkan aplikasi tidak akan bertahan lama. *Perspective* revisi produk akan melihat apakah sebuah aplikasi dapat bertahan jika ditambahkan fitur baru, dilakukan pemeliharaan, dan sebagainya. Pengujian nonfungsional dengan metode *McCall's Factor* pada *perspective* revisi produk yang dipilih hanya dapat dilakukan secara manual, karena hingga saat ini belum ditemukan *tools* untuk melakukan pengujian secara otomatis.

Ivent merupakan aplikasi berbasis web yang berguna untuk membantu masyarakat dalam mencari *event organizer*. Pengguna Ivent dapat memilih *event organizer* atau vendor yang sesuai dengan keinginan pengguna. Perangkat lunak berbasis web ini dikembangkan dengan metode *waterfall* yang dinilai mampu meminimalisir kesalahan pada pengembangan perangkat lunak [4]. Ivent dipilih sebagai kasus dalam penelitian karena aplikasi ini belum pernah dilakukan pengujian secara nonfungsional. Penelitian ini bertujuan untuk mengetahui cara pengujian perangkat lunak dengan menggunakan pendekatan *McCall's Factor* pada *perspective* revisi produk terhadap aplikasi Ivent. Adanya penelitian ini diharapkan *developer* dapat terbantu dalam memperbaiki kerusakan dan kekurangan yang terdapat pada perangkat lunak. Kualitas perangkat lunak juga dapat berkembang lebih baik dengan peningkatan performa implementasinya.

II. KAJIAN PUSTAKA

A. Pengujian Nonfungsional Perangkat Lunak

Menurut Chung dan Prado Leite *non-functional requirements* merupakan spesifikasi yang menggambarkan kemampuan dan kendala operasi sistem yang berguna untuk meningkatkan fungsionalitas [5]. Penelitian yang dilakukan oleh Fata Nidaul Khasanah menguji aplikasi telepon darurat

berbasis android secara nonfungsional [6]. Penelitian tersebut menggunakan metode BETA untuk pengujiannya. Pengujian tersebut dilakukan menggunakan survei dengan menyebarkan kuesioner yang berisikan penilaian pada setiap butir pertanyaan dengan menggunakan skala *likert* ke beberapa responden. Terdapat empat kriteria yang akan diuji dalam penelitian ini yaitu, *usefulness*, *ease of use*, *ease of learning*, dan *satisfaction.* Hasil dari survei diketahui bahwa nilai persentase yang diperoleh berdasarkan kriteria penilaian yaitu *usefulness* 81%, *ease of use* 80%, *ease of learning* 83%, *satisfaction* 81%. Dari nilai persentase tersebut, aplikasi telepon darurat berbasis android telah berjalan sesuai dengan hasil yang diharapkan. Berdasarkan penelitian yang telah dilakukan diketahui bahwa metode penelitian nonfungsional tersebut hanya menguji dari sisi pengguna dan tidak dari segala sisi. Oleh karena itu, tidak dapat diketahui fleksibilitas sistem tersebut jika diberikan fitur baru dan dilakukan pemeliharaan.

B. McCall's Factor

McCall mengklasifikasi tiga hal penting yang dapat memengaruhi kualitas perangkat lunak dari sudut pandang pengembangan produk yang dikenal sebagai perspektif produk (*product perspective*) [7]. Menurut *McCall* tiga hal penting dari perspektif produk seperti diperlihatkan pada Gambar 1 memiliki penjelasan sebagai berikut.

1. Transisi produk (*Product Transition*)

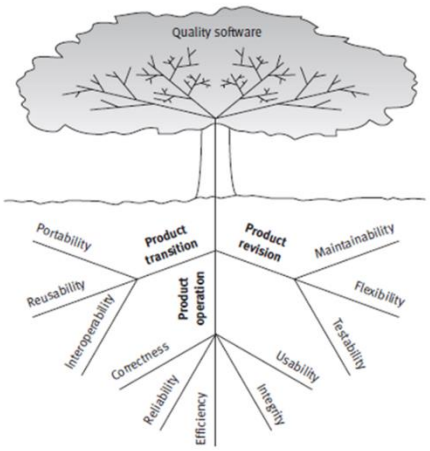
Kemampuan suatu produk untuk beradaptasi terhadap lingkungan baru. Bagaimana perangkat lunak dapat dijalankan pada platform yang beragam. Faktor yang berhubungan dengan transaksi produk adalah *portability*, *reusability*, *interoperability*.

2. Kinerja produk (*Product Operations*)

Kemampuan suatu produk yang berkaitan dengan karakteristik kinerja suatu aplikasi perangkat lunak. Faktor yang berhubungan dalam kinerja produk terdiri dari *integrity*, *correctness*, *usability*, *efficiency*.

3. Revisi produk (*Product Revision*)

Kemampuan suatu produk untuk mengalami sebuah perubahan. Perangkat lunak yang dirancang dan dikembangkan dengan baik, dapat dengan mudah diubah sesuai dengan kebutuhan. Revisi produk terdiri dari *maintainability*, *flexibility*, *testability*.



Gambar. 1.McCall's *Software Quality Factors* [8]

Rumus yang ditunjukkan pada Persamaan (1) digunakan untuk mengukur faktor kualitas suatu produk perangkat lunak. Penjelasan mengenai rumus pada Persamaan (1) adalah sebagai berikut.

$$Fq = W_1 * m_1 + W_2 * m_{2+\ldots\ldots+} W_n * m_n \quad (1)$$

di mana:

Fq = Faktor *software quality*

w_1 = Bobot yang tergantung pada produk dan kepentingannya

m_1 = Metrik yang memengaruhi faktor *software quality*

Nilai faktor *software quality* di atas akan diubah dalam bentuk persentase (%). Nilai persentase tersebut akan memberikan nilai untuk kualitas-kualitas yang diuji. Besarnya persentase dapat dihitung dengan rumus Persamaan (2) berikut [9].

$$\% = \frac{Rerata\ Faktor\ software\ quality}{Nilai\ Maksimum}\ x\ 100\% \ldots (2)$$

Pembagian kualitas dibagi menjadi lima, skala ini memperhatikan nilai dari persentase. Nilai maksimal yang diharapkan akan bernilai 100% dan nilai minimum sebesar 0%. Pembagian rentang kualitas dapat dilihat pada Tabel I seperti berikut.

TABLE I. RENTANG KATEGORI KUALITAS

Katergori	Presentase
Sangat Baik	81% - 100%
Baik	61% - 80%
Cukup Baik	41% - 60%
Tidak Baik	21% - 40%
Sangat Tidak Baik	<20%

Penelitian ini akan membahas pengujian pada perspektif *product revision.* Oleh karena itu, pembahasan berikutnya akan difokuskan pada *maintainability*, *flexibility*, dan *testability*.

C. Maintainability

Maintainability testing mengacu pada kemudahan dan kesulitan sebuah aplikasi yang diperlukan untuk membuat perubahan. *Maintainability testing* sangat diperlukan karena dengan menghitung nilai dari *maintainability* dapat membantu dalam memutuskan apakah suatu perangkat lunak mudah dirawat atau perlu perencanaan ulang [10]. Berdasarkan metode *McCall's Factor*, *Maintainability* ini dibagi menjadi lima *sub attributes* atau karakteristik. Karakteristik tersebut, yaitu *Conciseness*, *Self Descriptive*, *Modularity*, *Simplicity*, dan *Consistency.*

D. Flexibility

Flexibility testing merupakan pengujian dalam perangkat lunak yang digunakan untuk menggambarkan keseluruhan sistem. Hal ini biasanya mengacu pada kemampuan dari sebuah perangkat lunak untuk beradaptasi dengan kemungkinan atau perubahan di masa depan. Oleh karena itu, fleksibelitas sangat penting untuk menghasilkan

perangkat lunak yang berkualitas [11]. Menurut metode *McCall's Factor*, *flexibility testing* dilakukan pada fase pengujian yang memiliki subkarakteristik *Self Descriptive*, *Generality*, dan *Expandability*.

E. Testability

Testability testing merupakan upaya yang diperlukan untuk menguji suatu program agar memastikan program tersebut menjalankan fungsi dengan semestinya [12]. *Testability testing* berguna untuk mengukur kemampuan dari upaya yang diperlukan untuk menjalankan suatu program agar dapat melihat kinerja suatu program di lingkungan tertentu, serta secara aktif dapat memecahkan masalah yang ditemukan. Menurut metode *McCall's Factor*, *testability testing* dibagi menjadi beberapa *sub attribute*s atau karakteristik yaitu, *Self Descriptive, Modularity, Simplicity,* dan *Instrumentation.*

III. METODOLOGI

Penelitian ini memiliki enam tahapan proses pengujian yang harus dilaksanakan secara sistematis dan terencana pada aplikasi berbasis web Ivent. Pertama yaitu analisis kebutuhan, tahap ini *software requirement* yang didapatkan dari *stakeholder* akan dianalisis oleh tim QA. Tim QA akan menganalisis apa saja yang dapat diuji secara fungsional dan nonfungsional. Kemudian yaitu tahap perencanaan pengujian, tahapan ini tim QA menentukan rencana seperti menyiapkan *test case* berdasarkan faktor nonfungsional perangkat lunak. Selanjutnya tahap pengembangan kasus uji, tahapan ini menjadi acuan dalam pengujian berdasarkan *test case* yang telah dibuat pada tahap sebelumnya, di mana sistem yang diuji dapat memenuhi ketentuan dan standar tertentu.

Selanjutnya adalah tahapan pengaturan lingkungan uji, tahap ini menentukan syarat *software* yang diuji apakah dapat berjalan dengan baik dan benar-benar siap. Jika *software* sudah siap akan masuk ke dalam tahap selanjutnya yaitu eksekusi pengujian. Tahap ini merupakan pengujian yang dilakukan berdasarkan *test plan* dan *test case* yang telah dibuat pada tahapan sebelumnya. Kemudian tahap terakhir adalah penutupan siklus uji, tahap ini akan mengidentifikasi dan mengevaluasi seluruh tahapan yang telah dilakukan.

IV. HASIL DAN PEMBAHASAN

A. Analisis Kebutuhan

Penelitian ini melakukan analisis kebutuhan yang merupakan tahapan pertama *software testing* pada aplikasi Ivent. Kegiatan yang dilakukan pada fase ini berupa mengidentifikasi jenis tes yang dilakukan dan mengumpulkan detail prioritas *testing* mengenai apa saja yang dapat diuji secara fungsional dan nonfungsional hingga identifikasi lingkungannya [13]. Aplikasi Ivent ini merupakan aplikasi yang dapat membantu pengguna untuk bebas memilih dan menentukan *event organizer* atau vendor sesuai kebutuhan dengan mudah. Berbagai pemilik dari usaha *event organizer* dan vendor dikumpulkan pada satu platform. Berdasarkan tahapan analisis kebutuhan, penelitian ini akan menguji dengan melihat penulisan kode yang ditulis oleh *developer*, contohnya pembuatan akun pengguna, apakah email yang didaftarkan dapat dipastikan terlebih dahulu melalui verifikasi. Selain itu pengujian juga melihat dari dokumentasi desain pembuatan aplikasi web Ivent. Contohnya dengan melihat dari *activity diagram* dan *use case diagram*.

B. Perencanaan Pengujian

Aktivitas yang dilakukan pada tahapan ini adalah menentukan *test plan* sebelum pengujian dilakukan, agar pengujian lebih terstruktur prosesnya. *Test plan* dimulai dengan menentukan *test case* berdasarkan faktor nonfungsional perangkat lunak. Faktor-faktor tersebut akan terbagi menjadi beberapa karakteristik yang akan dicari nilainya, seperti *Maintainability* yang dibagi menjadi lima karakteristik, yaitu *Conciseness, Self Descriptive, Modularity, Simplicity,* dan *Consistency*. *Flexibility* yang terbagi menjadi tiga karakteristik, yaitu *Generality, Expandability*, dan *Self Descriptive.* Kemudian *Testability* yang terbagi menjadi empat karakteristik, yaitu *Self Descriptive, Modularity, Simplicity,* dan *Instrumentation*.

Perhitungan nilai akan menggunakan metrik yang berguna untuk mengukur keseluruhan sistem tentang bagaimana kemajuan sistem tersebut dari berbagai faktor kualitas. Jika nilai pada metrik rendah, hal tersebut akan memungkinkan adanya kegagalan, sehingga diperlukan tindakan korektif. Namun pada beberapa karakteristik nilai rendah tersebut mungkin juga tidak memiliki kegagalan secara umum. Melainkan metrik dapat mengetahui faktor mana yang bermasalah sehingga tindakan korektif dapat diaplikasikan untuk mengembangkan faktor tersebut agar lebih baik ke depannya. Perhitungan metrik tersebut dinilai berdasarkan *design,* dan *implementation*. Setelah mendapatkan nilai-nilai karakteristik, maka nilai tersebut akan dihitung menggunakan rumus *Software Quality McCall's Factor.* Untuk menghitung menggunakan *Software Quality McCall's Factor* dibutuhkan bobot yang bernilai sesuai dengan kepentingannya. Setelah berdiskusi dengan *developer*, maka bobot ini dianggap sama rata untuk setiap karakteristik yaitu 0.125. Hal ini untuk penyederhanaan kasus pengujian, semua karakteristik dalam penelitian ini dianggap penting. Tabel II merupakan penggunaan dari pembagian bobot berdasarkan kepentingannya dan *test case* yang akan digunakan.

TABLE II. TEST CASE METRIK

Kriteria	Bobot	Metric	Nilai	
			Des	Imp
Modularity	0.125	1. Stability Measure $\frac{Expected\ \#\ modules}{Total\ \#\ modules}$		
		2. Modular Implementation Measure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
Simplicity	0.125	1. Design Structure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
		2. Complexity Measure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
		3. Use Of Structured Language Or Preprocessor		

		1 jika menggunakan dan 0 jika tidak		
		4. Measure Of Coding Simplicity $1 - \frac{\#\ Of\ Elements}{\#executable\ statements}$		
Consisten-cy	0.125	1. Procedure Consistency $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
		2. Data Consistency Measure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
Generali-ty	0.125	1. Extent To which Module Is Referenced By Other Modules $\frac{\#common\ modules}{total\ \#\ modules}$		
		2. Implementation For Generality Checklist $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
Expanda-bility	0.125	1. Data Storage Expansion Measure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
		2. Extensibility Measure $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
Instrume n-tation	0.125	1. Module Testing Measure $\frac{\#path\ to\ be\ tested}{Total\ \#\ path}$		
		2. Integration Testing Measure $\frac{\#path\ to\ be\ tested}{Total\ \#\ Interface}$		
		3. System Testing Measure $\frac{\#modules\ to\ be\ executed}{Total\ \#\ of\ modules}$		
Concise-ness	0.125	1. Halstead's Measure $1 - \frac{Modul\ calculated - Modul\ Observed}{Module\ Observed}$		
Self Descrip-tive	0.125	1. Quantity Of Comments $\frac{\#\ Of\ Comments(non\ blank)}{Total\ \#\ Of\ Lines(non\ blank)}$		
		2. Effectiveness Of Comments $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		
		3. Descriptiveness Of Implementation Language $1 - \frac{\#modules\ violate\ rule}{Total\ \#\ modules}$		

C. Pengembangan Kasus Uji

- *Conciseness*

Karakteristik ini dihitung melalui *implementation* pada perangkat lunak. Aturan pada karakteristik akan menentukan apakah sistem dibagun dengan jumlah kode yang relatif singkat. Aturan ini dihitung menggunakan aturan *Halstead Measure*, didapatkan nilai 0,49, 0,53, 0,26, 0,64, 0,63, 0,50, 0,23, 0,54 dari perhitungan delapan modul. Kemudian dari nilai-nilai tersebut dirata-rata menggunakan rumus seperti Persamaan (3) berikut.

$$\frac{Sum\ Halstead's\ Measure\ for\ each\ module}{\text{Total \# module}} \quad (3)$$

$$\frac{0{,}49+0{,}53+0{,}26+0{,}64+0{,}63+0{,}50+0{,}23+0{,}54}{8} = 0{,}48$$

- *Self Descriptive*

Karakteristik ini dihitung melalui *implementation* pada perangkat lunak. Terdapat tiga aturan pada karakteristik ini yaitu, *Quantity of Comments*, *Effectiveness of Comments Measure*, *Descriptiveness of Implementation Language*. Aturan yang pertama didapatkan nilai 0,27, 0,45, 0,32, 0,33, 0,27, 0,11, 0,27, 0,24. Dari perhitungan delapan modul akan dirata-rata menggunakan rumus seperti Persamaan (4) berikut.

$$\frac{Penjumlahan\ Quantity\ of\ Comments\ Setiap\ Module}{\text{Total \# module}} \quad (4)$$

$$\frac{0{,}27+0{,}45+0{,}32+0{,}33+0{,}27+0{,}11+0{,}27+0{,}24}{8} = 0{,}28$$

Aturan yang kedua yaitu *Effectiveness of Comments Measure* memiliki tujuh aturan turunan dan didapatkan nilai 0, 1, 0, 0,13, 0, dari lima aturan turunan yang terpenuhi. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti Persamaan (5) berikut.

$$\frac{Total\ Score\ From\ Applicable\ Elements}{\text{\# Applicable Elements}} \quad (5)$$

$$\frac{0+1+0+0{,}13+0}{5} = 0{,}23$$

Aturan yang ketiga yaitu *Descriptiveness of Implementation Language* memiliki enam aturan turunan dan didapatkan nilai 0, 0,75, 0,75, 0, 0,16, 0. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0+0{,}75+0{,}75+0+0{,}16+0}{6} = 0{,}28$$

- *Modularity*

Karakteristik ini dihitung melalui *implementation* dan *design* perangkat lunak. Terdapat dua aturan pada karakteristik ini yaitu, *Stability Measure* dan *Modular Implementation*. Aturan yang pertama menghitung berdasarkan *design* tetapi tidak ditemukan dalam dokumentasi desain sehingga nilainya NA. Aturan yang kedua memiliki delapan aturan turunan dan didapatkan nilai 1, 0,88, 0, 1, 0,67, 1,1 dari tujuh aturan yang terpenuhi. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+0,88+0+1+0,67+1+1}{7} = 0,79$$

- *Simplicity*

Karakteristik ini dihitung melalui *implementation* dan *design* perangkat lunak. Karakteristik ini memiliki empat aturan yaitu, *design structure* yang akan dihitung melalui fase *implementation* dan *design*. Bagian *design* terdapat tujuh aturan turunan dan didapatkan nilai 1, 1, 0.33, 0, 0,67, 0.33 dari enam aturan. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+1+0.33+0+0,67+0.33}{6} = 0,56$$

Bagian *implementation* terdapat tujuh aturan turunan dan didapatkan nilai 1, 0, 0, 0, 1, 0 dari enam aturan. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+0+0+0+1+0}{6} = 0,33$$

Aturan selanjutnya *use of structured language or preprocessor* yang akan dihitung melalui fase *implementation*. Karakteristik ini terdapat satu aturan yang akan di uji pada kode yang ditulis oleh *developer*. Berdasarkan aturan tersebut didapatkan nilai 1, 1, 1, 1, 0, 0, 0, 1. Kemudian dari nilai-nilai tersebut dirata-rata menggunakan rumus seperti Persamaan (6) berikut.

$$\frac{Sum\ Of\ Modules\ Scores}{\#\text{Modules Scores}} \quad (6)$$

$$\frac{1+1+1+1+0+0+0+1}{8} = 0.63$$

Aturan berikutnya *complexity measure* yang dihitung melalui fase *design* dan *implementation*. Berdasarkan fase *design* terdapat satu aturan yang akan diuji berdasarkan dokumentasi yang dibuat oleh *developer*. Berdasarkan aturan tersebut didapatkan nilai 0,2 dan 0,8. Kemudian dari nilai-nilai tersebut dirata-rata menggunakan rumus seperti pada Persamaan (6).

$$\frac{0,2+0,8}{2} = 0.5$$

Berdasarkan fase *implementation* satu aturan yang akan diuji pada kode yang ditulis oleh *developer*. Berdasarkan aturan tersebut didapatkan nilai 0,4, 0,2, 0,14, 0,17, 0, 0,11, 0, 0,25. Kemudian dari nilai-nilai tersebut dirata-rata menggunakan rumus seperti pada Persamaan (6).

$$\frac{0,4+0,2+0,14+0,17+0+0,11+0+0,25}{8} = 0.16$$

Kemudian yang terakhir terdapat *measure of coding simplicity* yang akan dihitung melalui fase *implementation*. Pada aturan ini terdapat 16 aturan turunan yang didapatkan nilai 1, 1, 1, 1, 1, 1, 1, 1, 0,22, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.07, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,25, 0,24, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,32. Kemudian dari nilai-nilai tersebut dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{\begin{array}{c}1+1+1+1+1+1+1+1+0,22+1+0+0+0+0+0+0+0+0+0\\+0+0+0+0+0+0+0.07+1+1+1+1+1+1+1+1+1+1+1+1\\+1+1+1+1+1+1+1+1+1+1+1+1+1+0,5+0,24+1+0+1+0\\+0+1+1+1+1+1+0,32\end{array}}{16} = 0.26$$

Setelah mendapatkan rata-rata dari nilai tersebut, maka pada aturan ini akan dibagi dengan banyaknya modul yang *developer* buat dengan Persamaan (7) sebagai berikut.

$$\frac{sum\ of\ code\ simplicity\ measure\ for\ each\ module}{\#\text{Modules}} \quad (7)$$

$$\frac{0.26}{8} = 0.03$$

- *Consistency*

Karakteristik ini dihitung melalui *implementation* dan *design* perangkat lunak. Karakteristik ini memiliki dua aturan yaitu *Procedure Consistency Measure* dan *Data Consistency Measure*. Aturan pertama bagian *design* terdapat empat aturan turunan dan didapatkan nilai 1, 0,67 dari dua aturan turunan yang terpenuhi. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+0,67}{2} = 0,84$$

Bagian *implementation* terdapat tiga aturan turunan dan didapatkan nilai 0,67 dari satu aturan turunan yang terpenuhi. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0,67}{1} = 0,67$$

Aturan yang kedua yaitu *Data Consistency Measure*, bagian *design* memiliki lima aturan turunan dan didapatkan nilai 0,33, 0,33 dari dua aturan turunan yang terpenuhi. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0,33+0,33}{2} = 0,33$$

Bagian *implementation* memiliki empat aturan turunan dan didapatkan nilai 0,125, 1, 1, 1. Kemudian nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0,125+1+1+1}{4} = 0,78$$

- *Generality*

Karakteristik ini dihitung melalui *implementation* dan *design* perangkat lunak. Karakteristik ini memiliki dua aturan yaitu *Extent to Which Module is Referenced by Other Module* dan *Implementation For Generality Checklist*. Aturan pertama bagian *design* didapatkan nilai 0 karena fase dokumentasi tidak ada modul lain yang digunakan lebih dari satu. Bagian *implementation* didapatkan nilai 2 dari dua modul yang mengikuti aturan ini, kemudian dari nilai tersebut akan dirata-rata menggunakan rumus seperti Persamaan (8) berikut.

$$\frac{\#\ Common\ Modules}{\text{Total \#modules}} \quad (8)$$

$$\frac{2}{8} = 0{,}25$$

Aturan kedua bagian *design* didapatkan nilai 0,6, 0, 0 dari tiga aturan turunan. Kemudian dari nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0{,}6+0+0}{3} = 0{,}2$$

Bagian *implementation* didapatkan nilai 0, 0,5, 0,75 dari lima aturan turunan, kemudian dari nilai tersebut akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0+0{,}5+0{,}75}{3} = 0{,}42$$

- *Expandability*

Karakteristik ini dihitung melalui *implementation* dan *design* perangkat lunak. Karakteristik ini memiliki dua aturan yaitu *Data Storage Expansion Measure* dan *Extensibility Measure*. Aturan yang pertama bagian *design* didapatkan nilai 1 menggunakan rumus seperti Persamaan 5 di atas. Bagian *implementation* didapatkan nilai 1 dan 0 dari dua aturan turunan, kemudian dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+0}{2} = 0{,}5$$

Aturan kedua bagian *design* didapatkan nilai 0 karena dua aturan turunan tidak terpenuhi. Bagian *implementation* memiliki tiga aturan turunan dan didapatkan nilai 0, dan 1 dari dua aturan yang terpenuhi. Kemudian akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{0+1}{2} = 0{,}5$$

- *Instrumentation*

Karakteristik ini dihitung melalui *implementation* perangkat lunak. Terdapat tiga aturan turunan pada karakteristik ini yaitu, *Module Testing Measure*, *Integration Testing Measure*, *System Testing Measure*. Aturan yang dapat dihitung hanya aturan pertama yaitu *Module Testing Measure*, dua aturan lainnya bernilai NA karena pada dokumentasi desain tidak ditemukan cara pengujiannya. Aturan yang pertama memiliki dua aturan turunan dan didapatkan nilai 1 dan 0. Kemudian akan dirata-rata menggunakan rumus seperti pada Persamaan (5).

$$\frac{1+0}{2} = 0{,}5$$

Setelah mendapatkan rata-rata dari nilai tersebut, maka pada aturan ini akan dibagi dengan banyaknya modul yang *developer* buat dengan Persamaan (9) sebagai berikut.

$$\frac{Total\ Scores\ from\ module\ testing\ measure}{\text{Total \#Modules}} \quad (9)$$

$$\frac{0.5}{8} = 0{,}06$$

D. Pengaturan Lingkungan Uji

Tahap ini menentukan syarat *software* yang diuji apakah dapat berjalan dengan baik dan benar-benar siap. Dalam menentukan sebuah *software* dapat diuji dengan baik diperlukan tiga tahapan yang harus ditinjau dalam pembuatan *software* tersebut, yaitu ketika dalam menyiapkan *requirement*, *design*, dan *implementation*. Tahap *requirement, software* Ivent ini terdapat tiga *requirement* yang dapat dianalisis yaitu dari kebutuhan *input*, proses, dan *output*. Kebutuhan input membahas tentang fungsi yang ada pada perangkat lunak ditinjau dari tiga aktor, yaitu admin, konsumen, dan penjual. Kebutuhan proses membahas tentang perancangan dari perangkat lunak yang ada pada kebutuhan input, seperti pembuatan *use case diagram, activity diagram*, basis data perangkat lunak, *prototype* dari perangkat lunak, dan pengujian perangkat lunak. Kebutuhan output bertujuan untuk melihat hasil yang ditampilkan pada perangkat lunak apakah sudah sesuai dengan input yang diberikan atau belum.

Selanjutnya tahapan *design*, perancangan desain ini terdiri dari pembuatan *use case diagram, activity diagram*, perancangan basis data sesuai spesifikasi, dan perancangan antarmuka. Pembuatan *use case diagram* bertujuan untuk menggambarkan sebuah urutan aktivitas dan interaksi antara pengguna dengan perangkat lunak. *Activity diagram* berguna untuk menggambarkan aktivitas awal hingga akhir dari suatu proses [14]. Pembuatan rancangan basis data bertujuan untuk mempresentasikan hubungan antara satu tabel dengan tabel lainnya. Perancangan antarmuka berguna untuk model atau gambaran awal dari antarmuka perangkat lunak yang akan mempermudah implementasi perangkat lunak yang akan dibangun. Kemudian tahap *implementation*, tahapan ini merupakan tahapan setelah proses *requirement* dan *design*. Pada tahapan ini *developer* akan menuliskan *source code* untuk pengembangan aplikasi.

E. Eksekusi Pengujian

- Hasil Analisis Faktor Kualitas Implementasi Maintainability

$$Fq = W_1 n_1 + W_2 n_2 \ldots\ldots + W_n n_n$$
$$Conciseness = 0.125 * 0.48 = 0.06$$
$$Self\ Descriptive = 0.125 * 0.28 + 0.125 * 0.23 + 0.125 * 0.28 = 0.1$$
$$Modularity = 0.125 * 0.79 = 0.1$$
$$Simplicity = 0.125 * 0.33 + 0.125 * 0.63 + 0.125 * 0.16 + 0.125 * 0.03 = 0.14$$
$$Consistency = 0.125 * 0.67 + 0.125 * 0.78 = 0.18$$

$$Fq = 0.06 + 0.1 + 0.1 + 0.14 + 0.18 = 0.58$$

$$\% = \frac{Rerata\ Faktor\ software\ quality}{Nilai\ Maksimum}\ x\ 100\%$$

$$Rerata = \frac{0.58}{5} = 0.12$$

$$\% = \frac{0.12}{1}\ x\ 100\% = 12\%$$

- Hasil Analisis Faktor Kualitas Implementasi Flexibility

$$Self\ Descriptive = 0.125 * 0.28 + 0.125 * 0.23 + 0.125 * 28 = 0.1$$

$$Generality = 0.125 * 0.25 + 0.125 * 0.42 = 0.08$$
$$Expandability = 0.125 * 0.5 + 0.125 * 0.5 = 0.13$$

$$Fq = 0.1 + 0.08 + 0.13 = 0.31$$

$$Rerata = \frac{0.31}{3} = 0.10$$

$$\% = \frac{0.10}{1}\ x\ 100\% = 10\%$$

- Hasil Analisis Faktor Kualitas Implementasi Testability

$$Self\ Descriptive = 0.125 * 0.28 + 0.125 * 0.23 + 0.125 * 28 = 0.1$$
$$Modularity = 0.125 * 0.79 = 0.1$$
$$Simplicity = 0.125 * 0.33 + 0.125 * 0.63 + 0.125 * 0.16 + 0.125 * 0.03 = 0.14$$
$$Instrumentation = 0.125 * 0.06 = 0.01$$

$$Fq = 0.1 + 0.1 + 0.14 + 0.01 = 0.35$$

$$Rerata = \frac{0.35}{4} = 0.09$$

$$\% = \frac{0.09}{1}\ x\ 100\% = 9\%$$

- Hasil Analisis Faktor Kualitas Desain Maintainability

$$Simplicity = 0.125 * 0.56 + 0.125 * 0.5 = 0.13$$
$$Consistency = 0.125 * 0.84 + 0.125 * 0.33 = 0.15$$

$$Fq = 0.13 + 0.15 = 0.28$$
$$Rerata = \frac{0.28}{2} = 0.14$$

$$\% = \frac{0.14}{1}\ x\ 100\% = 14\%$$

- Hasil Analisis Faktor Kualitas Desain Flexibility

$$Generality = 0.125 * 0 + 0.125 * 0.2 = 0.025$$
$$Expandability = 0.125 * 1 + 0.125 * 0 = 0.125$$

$$Fq = 0.025 + 0.125 = 0.15$$

$$Rerata = \frac{0.15}{2} = 0.075$$

$$\% = \frac{0.075}{1}\ x\ 100\% = 7.5\%$$

- Hasil Analisis Faktor Kualitas Desain Testability

$$Simplicity = 0.125 * 0.56 + 0.125 * 0.5 = 0.13$$

$$Fq = 0.13$$

$$Rerata = \frac{0.13}{1} = 0.13$$

$$\% = \frac{0.13}{1}\ x\ 100\% = 13\%$$

F. Penutupan Siklus Uji

Nilai dari hasil pengujian perangkat lunak berbasis web Ivent masih kurang baik. Tabel III memperlihatkan hasil perhitungan faktor nonfungsional. Nilai *Software Quality* berdasarkan perhitungan fase implementasi untuk faktor *maintainability* sebesar 12%, faktor *flexibility* sebesar 10%, dan faktor terakhir yaitu *testability* sebesar 9%. Nilai dari hasil perhitungan fase desain untuk faktor *maintainability* sebesar 14%, faktor *flexibility* sebesar 7.5%, dan faktor terakhir yaitu *testability* sebesar 13%. Jika ditinjau dari Tabel 1 mengenai rentang kategori kualitas, nilai persentase yang didapatkan masih sangat tidak baik. Oleh karena itu, diperlukan perbaikan oleh tim *developer* untuk memelihara perangkat lunak agar lebih baik.

TABLE III. NILAI AKHIR KARAKTERISTIK

Faktor / **Karakteristik**	**Maintainability**		**Flexibility**		**Testability**	
	Des	**Imp**	**Des**	**Imp**	**Des**	**Imp**
Conciseness		0.06				
Self Descriptive		0.1		0.1		0.1
Modularity	NA	0.1			NA	0.1
Simplicity	0.13	0.14			0.13	0.14
Consistency	0.15	0.18				
Generality			0.025	0.08		
Expandability			0.125	0.13		
Instrumentation					NA	0.01
Rata-Rata	**14%**	**12%**	**7.5%**	**10%**	**13%**	**9%**
	13%		**8.75%**		**11%**	

V. KESIMPULAN

Penelitian ini difokuskan pada pengujian kualitas perangkat lunak nonfungsional. Berdasarkan hasil penelitian, pengujian perangkat lunak Ivent dengan menggunakan pendekatan *McCall's Factor* pada aspek *product revision* melalui tahapan awal dengan menentukan jenis tes yang dibutuhkan. Pengujian dilanjukan dengan menentukan metrik pengujian dan perumusan pada setiap karakteristik yang telah ditentukan. Kemudian, penelitian melakukan pengaturan lingkungan uji untuk menentukan syarat *software* telah benar-benar siap untuk diuji yang dilanjutkan dengan melakukan eksekusi pengujian.

Berdasarkan perhitungan implementasinya, didapatkan nilai *Software Quality* untuk faktor *Maintainability* sebesar 12%, faktor *Flexibility* sebesar 10%, dan *Testability* sebesar 9%. Sedangkan nilai dari hasil pengujian perangkat lunak pada fase desain untuk faktor *Maintainability* sebesar 14%, faktor *Flexibility* sebesar 7.5%, dan faktor terakhir yaitu *Testability* sebesar 13%. Perolehan nilai *Software Quality* tersebut dapat menjadi acuan bagi *developer* dalam melakukan perbaikan dan pengembangan kualitas perangkat lunak. Penelitian ini masih memiliki kekurangan, karena penelitian ini hanya berfokus pada faktor nonfungsional aspek *product revision*. Penelitian selanjutnya dapat diteruskan dengan menguji pada aspek *product transition* maupun *product operations* menggunakan metode *McCall's Factor*.

REFERENSI

[1] K. Naik and P. Tripathy, "Software Testing and Quality Assurance : Theory and Practice," Sep. 2011, Accessed: Jun. 29, 2023. [Online]. Available: https://books.google.co.id/books?id=neWaoJKSkvgC

[2] A. B. Pohan and M. Kom, "Pengujian dan Implementasi Sistem," 2018, Accessed: Jun. 29, 2023. [Online]. Available: https://scholar.google.co.id/citations?view_op=view_citation&hl=id&user=Peq20YgAAAAJ&citation_for_view=Peq20YgAAAAJ:8k81kl-MbHgC

[3] M. U. Siregar and A. H. Arif, "A Usage of McCall's Software Quality Analysis on the Bonus System of PT Surya Pratama Alam," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 3, no. 1, p. 63, Dec. 2018, doi: 10.14421/jiska.2018.31-07.

[4] M. Aikal, "Pengembangan E-commerce vendor dan Event Organizer Berbasis Website dengan Metode Waterfall," Aug. 2021, Accessed: Jun. 29, 2023. [Online]. Available: https://dspace.uii.ac.id/handle/123456789/34251

[5] L. Chung and J. C. S. do Prado Leite, "On Non-Functional Requirements in Software Engineering," in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 363–379. doi: 10.1007/978-3-642-02463-4_19.

[6] F. N. Khasanah, "Pengujian Fungsional Dan Non Fungsional Aplikasi Informasi Telepon Darurat Berbasis Android," *INFORMATION SYSTEM FOR EDUCATORS AND PROFESSIONALS*, vol. 3, no. 1, pp. 79–90, 2018.

[7] J. McCall, P. Richards, and G. Walters, "Factors in Software Quality," *Tehnical Report RADC-TR-77-369*, vol. 1, Nov. 1977.

[8] T. Weerasinghe, "Software Quality Factors," *Medium*. Jun. 2022. [Online]. Available: https://thilinaweerasinghe.medium.com/software-quality-factors-c0f57bbfc3d2

[9] A. Prayitno and Y. Safitri, "Pemanfaatan Sistem Informasi Perpustakaan Digital Berbasis Website Untuk Para Penulis," *Indonesian Journal on Software Engineering*, vol. 1, no. 1, 2015.

[10] V. Sharma and P. Baliyan, "Maintainability Analysis of Component Based Systems," *International Journal of Software Engineering and Its Applications*, vol. 5, no. 3, 2011.

[11] K. M. Nelson, H. J. Nelson, and M. Ghods, "Technology flexibility: Conceptualization, validation, and measurement," in *Proceedings of the Hawaii International Conference on System Sciences*, 1997, pp. 76–87. doi: 10.1109/HICSS.1997.661572.

[12] J. Mona, "Software Testability (Its Benefits, Limitations, and Facilitation)," vol. 445, pp. 287–298, Sep. 2022, doi: https://doi.org/10.1007/978-981-19-1412-6_23.

[13] A. Aulia Aziiza and A. N. Fadhilah, "Analisis Metode Identifikasi dan Verifikasi Kebutuhan Non Fungsional," *Applied Technology and Computing Science Journal*, vol. 3, no. 1, 2020.

[14] U. Tanoto, "Activity Diagram: Pengertian, Fungsi, Contoh serta Cara Membuatnya," *Jojonomic Aplikasi HRIS, Human Capital & Expense Management*. Dec. 2020. [Online]. Available: https://www.jojonomic.com/blog/activity-diagram/