BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem informasi adalah komponen yang saling berhubungan, mengumpulkan atau mendapat, memroses, menyimpan, dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan suatu organisasi [4]. Sistem informasi adalah suatu sistem yang di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan [5].

Sistem informasi adalah sekumpulan orang, prosedur, dan sumber daya yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi. Sstem informasi dalam suatu organisasi dapat dikatakan sebagai suatu sistem yang menyediakan informasi bagi semua tingkatan dalam organisasi tersebut kapan saja diperlukan. Dan juga bisa dikatakan sebagai sebuah sistem yang menerima sumber data sebagai masukan dan memroses mereka menjadi produk informasi sebagai keluaran [6].

Komponen-komponen sistem informasi adalah sebagai berikut [5]:

1. *Input*

   Input mewakili data yang masuk ke dalam sistem informasi. *Input* yang dimaksud adalah metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Model

   Model terdiri dari kombinasi prosedur, logika, dan model matematik yang akan memanipulasi data *input* dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. *Output*

   Produk dari sistem ini adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4. Teknologi

   Teknologi merupakan "*tool box*" dalam sistem informasi. Teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran, serta membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 (tiga) bagian utama, yaitu teknisi (*brainware*), perangkat lunak (*software*), dan perangkat keras (*hardware*).

5. Basis Data

   Basis data (*database*) merupakan kumpulan dari data yang saling berkaitan dan berhubungan satu sama lain, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan persediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpanannya. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut *Database Management System* (DBMS).

6. Kontrol

   Banyak hal yang dapat merusak sistem informasi, seperti bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan dari sistem itu sendiri, ketidakefisienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

2.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah teknik disiplin yang berkaitan dengan semua aspek produksi, dimulai dari tahap awal spesifikasi sistem hingga tahap pemeliharaan setelah perangkat telah digunakan [7].

Rekayasa adalah tentang mendapatkan hasil dari kualitas yang diperlukan sesuai jadwal dan anggaran. Terkadang hal ini tidak berjalan sesuai harapan. Secara umum, para *software engineer* mengadopsi pendekatan yang sistematis dan terorganisir untuk pendekatan saat bekerja. Hal ini merupakan cara yang paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun, rekayasa adalah

tentang memilih metode yang paling tepat untuk keadaan yang terjadi. Jadi, pendekatan yang kreatif kurang tepat untuk membangun beberapa jenis perangkat lunak [7].

Proses yang fleksibel merupakan metode modifikasi perubahan yang sangat sesuai untuk pengembangan interaktif bagi sistem yang berbasis *web* dan aplikasi mobil, yang membutuhkan perpaduan antara perangkat lunak dan grafik. Rekayasa perangkat lunak penting karena memiliki dua alasan, yaitu [7]:

1. Semakin banyak individu ataupun masyarakat yang bergantung pada sistem perangkat lunak yang canggih, maka dibutuhkan pula untuk dapat menghasilkan sistem yang handal dan dapat dipercaya secara ekonomis dan cepat.
2. Biasanya lebih murah dalam jangka panjang untuk menggunakan metode rekayasa perangkat lunak dan teknik untuk sistem perangkat lunak daripada hanya menulis program sebagai proyek pemrograman individu.

Gagal dalam menggunakan rekayasa perangkat lunak berpengaruh terhadap biaya yang lebih tinggi untuk pengujian, jaminan kualitas, dan pemeliharaan jangka panjang. Sebagai *software engineer* harus melibatkan tanggung jawab yang lebih luas daripada hanya penerapan keterampilan teknis. Seorang *software engineer* juga harus memiliki etika dan cara yang bertanggung jawab secara moral jika ingin dihormati sebagai seorang yang profesional. Seorang *software engineer* tidak seharusnya menggunakan kemampuan dan keterampilan yang dimiliki dengan cara yang tidak jujur atau dengan cara yang membawa keburukan pada profesi itu sendiri [7].

Beberapa etika yang harus dimiliki oleh seorang *software engineer* adalah sebagai berikut [7]:

1. *Confidentiality*. Menjaga kerahasiaan dan menghormati klien, terlepas apakah perjanjian kerahasiaan itu formal ataupun tidak formal.
2. Kompetensi (*Competence*). Tidak boleh salah menggambarkan tingkat kompetensi, harus secara sadar menerima pekerjaan yang berada di luar kompetensi.
3. Hak Kekayaan Intelektual (*Intellectual Property Rights*). Harus mengetahui hukum setempat yang mengatur penggunaan kekayaan intelektual seperti hak paten dan hak cipta, harus hati-hati untuk memastikan bahwa kekayaan intelektual dari pemberi kerja dan klien dilindungi.

2.3 Pengujian Perangkat Lunak

Pengujian dimaksudkan untuk menunjukkan bahwa suatu program melakukan apa yang seharusnya dilakukan, dan hal ini dilakukan untuk menemukan kecacatan program sebelum mulai digunakan. Saat menguji perangkat lunak harus mengeksekusi sebuah program yang menggunakan data buatan, memeriksa hasil tes untuk kesalahan, anomali, atau informasi tentang atribut nonfungsional program [7].

Saat melakukan pengujian perangkat lunak, ada dua hal yang harus dilakukan [7]:

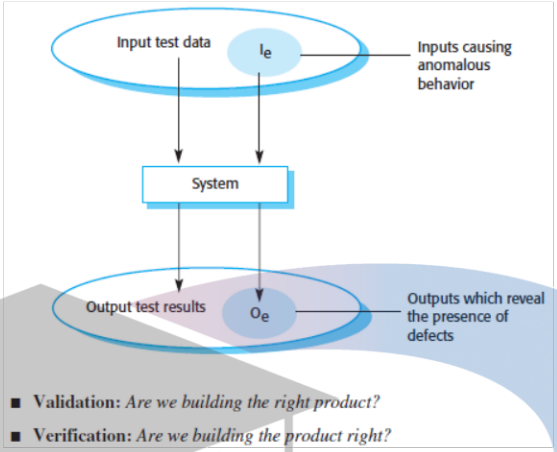
1. Tunjukkan kepada pengembang dan pelanggan bahwa perangkat lunaknya memenuhi persyaratan. Untuk perangkat lunak khusus, ini berarti setidaknya harus ada satu uji untuk setiap persyaratan dalam dokumen persyaratan. Untuk perangkat lunak produk generik, itu berarti harus ada tes untuk semua fitur sistem untuk dimasukkan dalam rilis produk. Dapat juga menguji kombinasi fitur untuk memeriksa interaksi yang tidak diinginkan di antara mereka.
2. Temukan *input* atau urutan *input* dimana perilaku perangkat lunak tidak benar, tidak diinginkan, atau tidak sesuai dengan spesifikasinya. Ini disebabkan oleh cacat (*bug*) dalam perangkat lunak. Ketika menguji perangkat lunak untuk menemukan cacat, berusaha membasmi perilaku sistem yang tidak diinginkan, seperti *crash* sistem, interaksi yang tidak diinginkan dengan sistem lain, perhitungan yang salah, dan kehilangan data.

Pengujian adalah bagian dari proses verifikasi dan validasi perangkat lunak (V&V) yang lebih luas. Verifikasi dan validasi bukan hal yang sama, walaupun seringkali membingungkan. Barry Boehm, seorang pelopor rekayasa perangkat lunak, secara ringkas menyatakan perbedaannya [7].

Proses verifikasi dan validasi berkaitan dengan memeriksa perangkat lunak yang sedang dikembangkan memenuhi spesifikasinya dan memberikan fungsionalitas yang diharapkan oleh orang-orang yang membayar perangkat lunak. Proses pengecekan ini dimulai segera setelah persyaratan menjadi tersedia dan melanjutkan semua tahapan proses pengembangan [7].

Proses verifikasi dan validasi berkaitan dengan memeriksa perangkat lunak apakah sesuai dengan spesifikasi dan memberikan fungsionalitas yang diharapkan oleh seorang *client*. Proses pengecekan ini dimulai segera setelah persyaratan menjadi tersedia dan melanjutkan semua tahapan proses [7].

Gambar berikut ini menunjukkan perbedaan di antara verifikasi dan validasi [7].



Gambar 2.1 *Input Output Model* untuk *Program Testing*

2.4 Metode McCall

Teori kualitas McCall merupakan model pengujian yang tertua, dikembangkan pada tahun 1976. Model ini pertama kali digunakan untuk sebuah implementasi proyek besar dalam US Air Force. Model ini bertujuan untuk menjembatani jarak antara *user* dan *developer*. Yang melatarbelakangi model ini adalah karena kurang jelasnya kebutuhan yang ditetapkan. Mencakup aspek penting dari fungsional sebuah *software* adalah penyebab dari buruknya performa suatu *software* [8].

Faktor-faktor yang memengaruhi kualitas perangkat lunak dapat dikategorikan dalam 2 (dua) kelompok besar, yaitu [8]:

1. Faktor yang dapat diukur secara langsung (misalnya cacat per titik fungsi), dan
2. Faktor yang dapat diukur hanya secara tidak langsung (misalnya kegunaan atau pemeliharaan), yaitu dengan membandingkan perangkat lunak (dokumen, program, data) ke beberapa informasi dan sampai pada indikasi kualitas.

Untuk membuat *software* yang memiliki performa baik, maka pada saat inisiasi harus menggali kebutuhan dari pengguna secara tepat. McCall dan kawan-kawan pada tahun 1977 telah mengusulkan suatu penggolongan faktor-faktor atau kriteria yang mempengaruhi kualitas *software*. McCall, Richards, dan Walters mengusulkan kategorisasi faktor yang berguna yang memengaruhi kualitas perangkat lunak. Faktor kualitas perangkat lunak berfokus pada tiga aspek penting dari produk perangkat

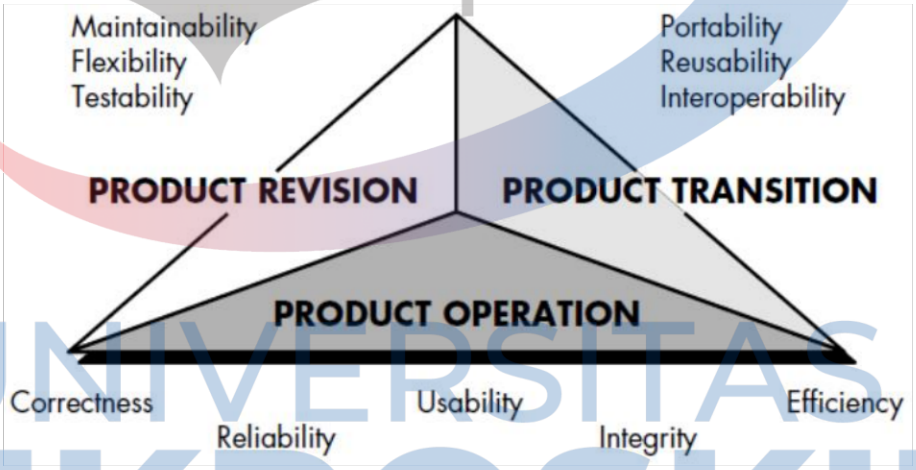
lunak, yaitu karakteristik operasionalnya, kemampuan untuk mengalami perubahan, dan kemampuan beradaptasi dengan lingkungan yang baru [8].

2.4.1 Faktor-Faktor Kualitas Menurut McCall

McCall menitikberatkan faktor-faktor kualitas menjadi tiga aspek penting, yaitu yang berhubungan dengan [8]:

1. Sifat-sifat operasional dari *software* (*Product Operations*).
2. Kemampuan *software* dalam menjalani perubahan (*Product Revision*).
3. Daya adaptasi atau penyesuaian *software* terhadap lingkungan baru (*Product Transition*).

Gambar berikut ini menunjukkan tiga aspek penting faktor-faktor kualitas perangkat lunak menurut McCall [8].



Gambar 2.2 Faktor Kualitas Perangkat Lunak McCall

Berdasarkan gambar di atas, penjelasan mengenai faktor-faktor kualitas menurut McCall adalah sebagai berikut [8]:

1. Kebenaran (*correctness*): Bagaimana program akan memberikan hasil sesuai dengan spesifikasi yang telah ditetapkan sebelumnya dan memenuhi sasaran-sasaran pelanggan.
2. Keandalan (*reliability*): Bagaimana suatu program diharapkan dapat melakukan fungsi-fungsi tertentu sesuai dengan tingkat ketelitian yang diinginkan.
3. Efisiensi (*efficiency*): Jumlah sumber daya komputasi dan kode yang diperlukan program untuk mampu melaksanakan fungsinya secara baik dan benar.

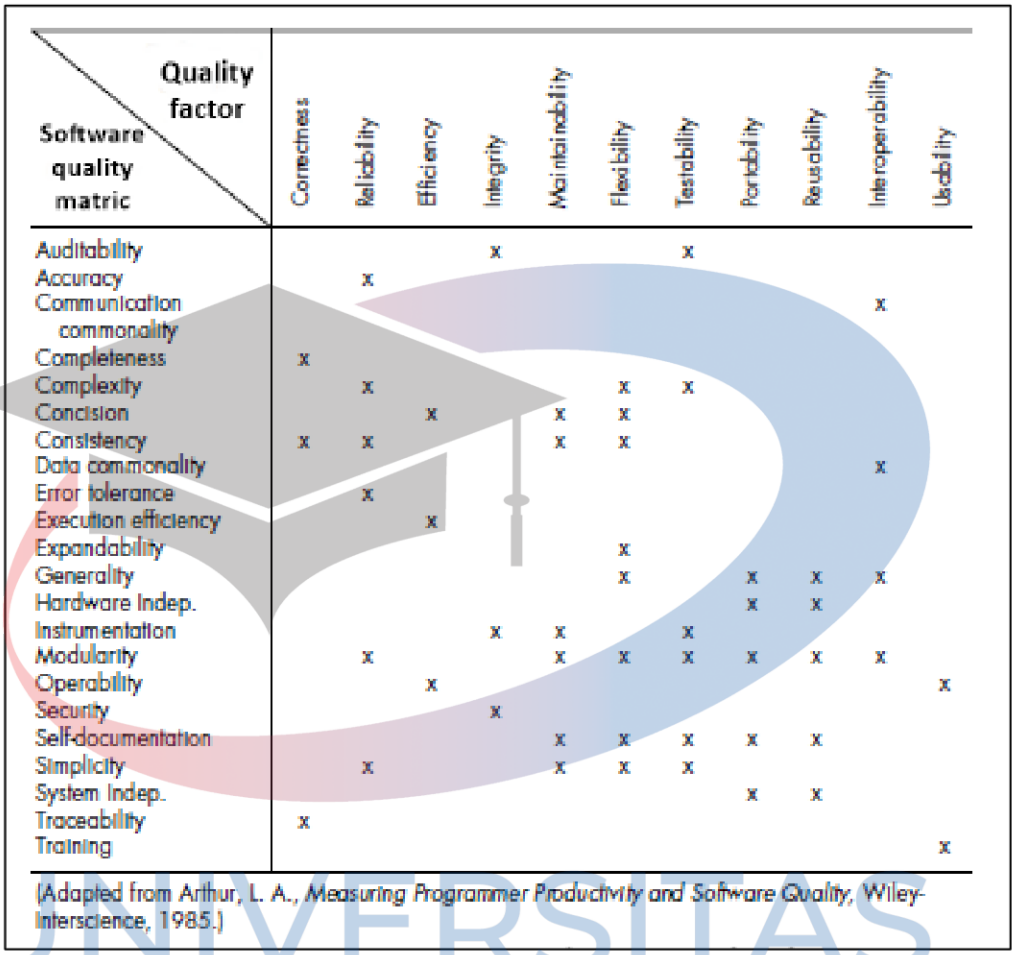
4. Integritas (*integrity*): Bagaimana akses ke perangkat lunak atau ke data oleh orang-orang yang tidak terotorisasi dapat dikendalikan.
5. Penggunaan (*usability*): Besarnya usaha yang diperlukan untuk mempelajari, mengoperasikan, menyediakan asupan (*input*), dan menafsirkan luaran (*output*) untuk suatu program.
6. Kemampuan untuk dipelihara (*maintainability*): Besarnya usaha yang diperlukan untuk melokalisasi dan membetulkan kesalahan-kesalahan yang dapat ditemukan dalam program.
7. Fleksibilitas (*flexibility*): Besarnya usaha yang diperlukan untuk memodifikasi suatu program yang bersifat operasional.
8. Kemampuan untuk menghadapi pengujian (*testability*): Besarnya usaha yang diperlukan untuk melakukan pengujian atas suatu program dengan tujuan untuk memastikan bahwa program itu melaksanakan fungsi yang diharapkan.
9. Portabilitas (*portability*): Besarnya usaha yang diperlukan untuk mentransfer program dari suatu perangkat keras dan/atau lingkungan perangkat lunak sistem ke perangkat keras dan/atau lingkungan perangkat lunak sistem lainnya.
10. Penggunaan ulang (*reusability*): Bagaimana suatu program (atau bagian suatu program) dapat digunakan ulang di aplikasi/program yang lainnya yang berhubungan dengan pengemasan dan lingkup fungsi-fungsi yang dilakukan oleh aplikasi/program.
11. Interoperabilitas (*interoperability*): Besarnya usaha yang diperlukan untuk menggantikan bagian suatu sistem dengan bagian sistem yang lainnya.

Terdapat beberapa metrik yang digunakan untuk mengukur kuantitas dari kualitas perangkat lunak yang dikembangkan berdasarkan pembagian yang diajukan oleh McCall mengenai formula untuk mengukur faktor-faktor kualitas perangkat lunak. Metrik-metrik berikut ini digunakan dalam pengukuran tersebut [8]:

1. Audibilitas (*auditability*): Kecocokan dimana keselarasan terhadap standar dapat diperiksa.
2. Akurasi (*accuracy*): Ketelitian komputasi dan kontrol.
3. Kelaziman komunikasi (*communication commonality*): Tingkat dimana *interface* standar, protokol, dan *bandwidth* digunakan.

4. Kelengkapan (*completeness*): Derajat dimana implementasi penuh dari fungsi yang diharapkan telah tercapai.
5. Keringkasan (*conciseness*): Kepadatan program dalam bentuk baris kode.
6. Konsistensi (*consistency*): Penggunaan desain dan teknik dokumentasi yang seragam pada keseluruhan proyek pengembangan perangkat lunak.
7. Kelaziman data (*data commonality*): Penggunaan struktur dan tipe data standar pada seluruh program.
8. Toleransi kesalahan (*error tolerance*): Kerusakan yang terjadi pada saat program mengalami kesalahan.
9. Efisiensi eksekusi (*execution efficiency*): Kinerja *run-time* dari suatu program.
10. Ekspandibilitas (*expandability*): Tingkat dimana arsitektur, data, atau desain prosedural dapat diperluas.
11. Generalitas (*generality*): Luas aplikasi potensial dari komponen program.
12. Independensi perangkat keras (*hardware independence*): Tingkat dimana perangkat lunak dipisahkan dari perangkat keras tempat ia beroperasi.
13. Instrumentasi (*instrumentation*): Tingkat dimana program memonitor operasinya sendiri dan menentukan kesalahan yang terjadi.
14. Modularitas (*modularity*): Independensi fungsional dari komponen program.
15. Operabilitas (*operability*): Kecocokan operasi program.
16. Keamanan (*security*): Availabilitas mekanisme yang mengontrol atau melindungi program dan data.
17. Pendokumentasian diri (*self-documentation*): Tingkat dimana kode sumber memberikan dokumentasi yang berguna.
18. Kesederhanaan (*simplicity*): Tingkat dimana sebuah program dapat dipahami tanpa kesukaran.
19. Independensi sistem perangkat lunak (*software system independence*): Tingkat dimana program tidak tergantung pada bentuk bahasa pemrograman nonstandar, karakteristik sistem operasi, dan batasan lingkungan yang lain.
20. Pelacakan (*traceability*): Kemampuan untuk menelusur-balik suatu representasi desain atau komponen program aktual ke persyaratan.
21. Pelatihan (*training*): Tingkat dimana perangkat lunak memungkinkan pemakai baru untuk mengaplikasikan sistem.

Hubungan antara faktor kualitas dan metrik kualitas perangkat lunak ditunjukkan pada gambar berikut ini [8].



Software quality matric \ Quality factor	Correctness	Reliability	Efficiency	Integrity	Maintainability	Flexibility	Testability	Portability	Reusability	Interoperability	Usability
Auditability				x			x				
Accuracy		x									
Communication commonality										x	
Completeness	x										
Complexity		x				x	x				
Concision			x		x	x					
Consistency	x	x			x	x					
Data commonality										x	
Error tolerance		x									
Execution efficiency			x								
Expandability						x					
Generality						x		x	x	x	
Hardware indep.								x	x		
Instrumentation				x	x		x				
Modularity		x			x	x	x	x	x	x	
Operability			x								x
Security				x							
Self-documentation					x	x	x	x	x		
Simplicity		x			x	x	x				
System indep.								x	x		
Traceability	x										
Training											x

(Adapted from Arthur, L. A., *Measuring Programmer Productivity and Software Quality*, Wiley-Interscience, 1985.)

Gambar 2.3 Hubungan Antara Faktor Kualitas dan Metrik Kualitas Perangkat Lunak

Faktor kualitas yang dijelaskan oleh McCall dan rekan-rekannya mewakili satu dari sejumlah "daftar periksa" yang disarankan untuk kualitas perangkat lunak. Hewlett-Packard mengembangkan seperangkat faktor kualitas perangkat lunak yang disebut FURPS: fungsionalitas (*functionality*), kegunaan (*usability*), keandalan (*realibility*), kinerja (*performance*), dan dukungan (*supportability*) [8].

Faktor kualitas FURPS menggambarkan secara bebas dan mendefinisikan atribut berikut untuk masing-masing dari kelima faktor utama [8]:

1. Fungsionalitas dinilai dengan mengevaluasi *set* fitur dan kemampuan program, keumuman fungsi yang disampaikan, dan keamanan sistem keseluruhan.
2. Kegunaan dinilai dengan mempertimbangkan faktor manusia, estetika keseluruhan, konsistensi, dan dokumentasi.

3. Keandalan dievaluasi dengan mengukur frekuensi dan tingkat keparahan kegagalan akurasi hasil keluaran, *mean-time-to-failure* (MTTF), kemampuan untuk pulih dari kegagalan, dan prediktabilitas program.
4. Kinerja diukur dengan kecepatan pemrosesan, waktu respon, konsumsi sumber daya, *throughput*, dan efisiensi.
5. Dukungan menggabungkan kemampuan untuk memperluas program (ekstensibilitas), kemampuan beradaptasi, dan kemudahan servis (ketiga atribut ini mewakili istilah yang lebih umum, yaitu kemampuan dipelihara). Selain itu, juga termasuk kemampuan untuk menghadapi pengujian, kompatibilitas, kemampuan dikonfigurasi (kemampuan untuk mengatur dan mengontrol elemen-elemen dari konfigurasi perangkat lunak, kemudahan sistem dapat diinstalasi, dan kemudahan dimana masalah bisa dilokalisasi.

Faktor kualitas dan atribut FURPS dapat digunakan untuk membangun metrik kualitas untuk setiap langkah dalam proses rekayasa perangkat lunak [8].

2.4.2 Teknik Pengukuran McCall

Untuk menghitung kualitas dengan menggunakan metode McCall dapat menggunakan persamaan berikut ini [8].

$$Fa = w1 * c1 + w2 * c2 + w3 * c3 + \ldots + wn * cn \quad (2.1)$$

Dimana:

Fa = Nilai total dari faktor *a*

w = Bobot yang bergantung pada produk dan kepentingan

c = Metrik yang mempengaruhi faktor *software quality*

Sistem penilaian menggunakan tahapan sebagai berikut [9]:

1. Menentukan kriteria yang digunakan untuk mengukur suatu faktor
2. Menentukan bobot (w) dari setiap kriteria berdasarkan tingkat kepentingan
3. Menentukan skala dari nilai kriteria
4. Memberikan nilai pada setiap kriteria
5. Menghitung nilai total dengan rumus pada persamaan (2.1)
6. Mengubah nilai faktor kualitas dalam bentuk persentase (%) dengan menggunakan persamaan berikut.

$$Persentase = \frac{Nilai\ yang\ didapat}{Nilai\ maksimum}\ x\ 100\% \quad (2.2)$$

Hasil persentase digunakan untuk memberikan jawaban atas kelayakan dari aspek-aspek yang diteliti. Pembagian kategori kualitas ada lima. Skala ini memperlikatkan rentang dari bilangan persentase. Nilai maksimal yang diharapkan adalah 100% dan minimum 0%. Pembagian rentang kategori kualitas dapat dilihat pada tabel berikut ini [2].

Tabel 2.1 Kategori Kelayakan

Kategori	Persentase
Sangat Baik	81% s.d. 100%
Baik	61% s.d. 80%
Cukup Baik	41% s.d. 60%
Tidak Baik	21% s.d. 40%
Sangat Tidak Baik	< 21%