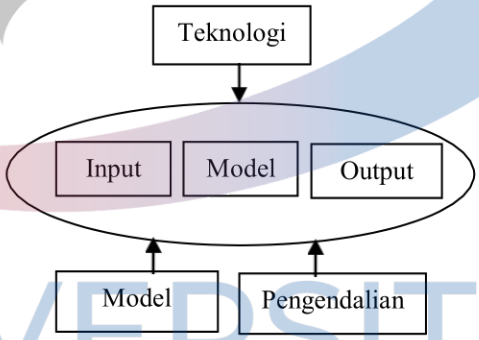
BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Informasi adalah data yang telah diolah menjadi bentuk yang lebih berguna bagi yang menerima. Sistem informasi adalah kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data, memroses dan menyimpan, serta mendistribusikan informasi [4].

Gambar berikut ini menunjukkan komponen-komponen dari sebuah sistem informasi [4].



Gambar 2.1 Komponen Sistem Informasi

Sistem informasi adalah seperangkat komponen terintegrasi untuk mengumpulkan, menyimpan, dan memroses data, serta untuk menyediakan informasi, pengetahuan, dan produk digital. Perusahaan bisnis dan organisasi lain bergantung pada sistem informasi untuk melaksanakan dan mengelola operasi, berinteraksi dengan pelanggan dan pemasok, serta bersaing di pasar. Sistem informasi digunakan untuk menjalankan rantai pasokan antar organisasi dan pasar elektronik. Sebagai contoh, perusahaan menggunakan sistem informasi untuk memroses akun keuangan, untuk mengelola sumber daya manusia, dan untuk menjangkau pelanggan potensial dengan promosi *online*. Individu bergantung pada sistem informasi, umumnya berbasis internet, untuk melakukan sebagian besar kehidupan pribadi mereka: untuk bersosialisasi, belajar, berbelanja, perbankan, dan lainnya. Sistem informasi mendukung operasi, pekerjaan,

pengetahuan, dan manajemen dalam organisasi [3].

Di dalam sistem informasi, manusia berinteraksi dengan manusia, manusia berinteraksi dengan komputer, dan komputer berinteraksi dengan komputer lain. Di dalam sistem informasi, data, informasi dan/atau pengetahuan mengalir dibawa oleh dokumen atau media komunikasi elektronik, seperti telepon atau jaringan komputer. Keberadaan sistem informasi diperlukan organisasi untuk mendampingi proses- proses bisnis dari organisasi. Contohnya, proses penjualan *supermarket* didampingi oleh sistem informasi penjualan, yang mencatat pengumpulan data dan informasi tentang penjualan [5].

2.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan. Pada definisi ini, ada dua istilah kunci, yaitu [6]:

1. Disiplin rekayasa: Perekayasa membuat suatu alat bekerja. Menerapkan teori, metode, dan alat bantu yang sesuai. Selain itu, menggunakannya dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung. Perekayasa juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan-batasan ini.
2. Semua aspek produksi perangkat lunak: Rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak, tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode, dan teori untuk mendukung produksi perangkat lunak.

Secara umum, rekayasa perangkat lunak memakai pendekatan sistematis dan terorganisasi terhadap pekerjaan mereka, karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun demikian, rekayasa ini sebenarnya mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan. 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 [6].

Proses perangkat lunak adalah serangkaian kegiatan-kegiatan dan hasil-hasil relevannya yang menghasilkan perangkat lunak. Kegiatan-kegiatan ini sebagian besar dilakukan perekayasa perangkat lunak. Ada empat kegiatan proses dasaryang umum bagi seluruh kegiatan proses perangkat lunak. Kegiatan-kegiatan ini adalah [7]:

1. Spesifikasi perangkat lunak, fungsionalitas perangkat lunak, dan batasan kemampuan operasinya harus didefinisikan.
2. Pengembangan perangkat lunak, perangkat lunak yang memenuhi spesifikasi tersebut harus diproduksi.
3. Validasi perangkat lunak, perangkat lunak harus divalidasi untuk menjamin bahwa perangkat lunak melakukan apa yang diinginkan oleh pelanggan.
4. Evolusi perangkat lunak, perangkat lunak harus berkembang untuk memenuhi kebutuhan pelanggan yang berubah-ubah.

Proses perangkat lunak yang berbeda mengatur kegiatan ini dengan cara berbeda dan dijelaskan dengan tingkat kerincian yang berbeda pula. Waktu kegiatan bervariasi, sebagaimana hasilnya. Pegaturan yang berbeda dapat menggunakan proses yang berbeda untuk menghasilkan produk dengan jenis yang sama. Namun demikian, untuk beberapa jenis aplikasi tertentu, beberapa proses lebih sesuai dari yang lainnya. Jika digunakan proses yang tidak sesuai, maka kualitas penggunaan produk perangkat lunak yang akan dikembangkan tersebut mungkin berkurang [6].

2.3 Pengujian Perangkat Lunak

Tahap pengujian konponen berhubungan dengan pengujian berfungsinya komponen yang teridentifikasi dengan jelas. Komponen ini dapat berupa fungsi atau sekumpulan metode yang digabungkan menjadi modul atau objek. Pada saat pengujian integrasi, komponen diintegrasikan untuk membentuk subsistem atau sistem yang lengkap. Pada tahap ini, pengujian harus terfokus pada interaksi antara komponen dan fungsionalitas serta kinerja sistem sebagai satu kesatuan.

Bagaimanapun tak akan terelakkan lagi bahwa cacat komponen yang terlewat pada pengujian awal akan ditemukan pada saat pengujian integrasi. Ketika menguji sistem kritis, spesifikasi rinci setiap komponen perangkat lunak digunakan oleh tim independen untuk membuat uji bagi sistem tersebut. Namun demikian, pada sebagian besar kasus lain, pengujian merupakan proses yang lebih intuitif karena tidak ada waktu untuk menulis spesifikasi rinci mengenai setiap bagian sistem perangkat lunak [6].

Berikut ini merupakan gambaran dari sebuah fase pengujian perangkat lunak [6].

Gambar 2.2 Fase Pengujian

Didasarkan atas pengalaman praktis ketika sistem dikembangkan dengan menggunakan model fungsional, berikut ini adalah jenis-jenis pengujian [6]:

1. Pengujian Cacat

   Tujuan pengujian cacat (*defect testing*) adalah mengungkap cacat laten pada sistem perangkat lunak sebelum sistem diserahkkan. Ini berlawanan dengan pengujian validasi yang ditunjukkan untuk mendemonstrasikan bahwa sistem telah memenuhi spesifikasinya. Pengujian validasi menuntut sistem berlaku dengan benar dengan menggunakan kasus uji penerimaan. Uji cacat yang berhasil merupakan uji yang menyebabkan sistem berlaku tidak benar dan dengan demikian mengungkap adanya cacat. Hal ini menekankan fakta penting mengenai pengujian.

2. Pengujian Kotak Hitam

   Pengujian fungsional atau pengujian kotak hitam (*blackbox testing*) merupakan pendekatan pengujian yang ujinya diturunkan dari spesifikasi program atau komponen. Sistem merupakan ”kotak hitam” yang perilakunya hanya dapat ditentukan dengan mempelajari *input* dan *output* yang berkaitan.

3. Pengujian Struktural

   Pengujian struktural (*structural testing*) merupakan pendekatan terhadap pengujian yang diturunkan dari pengetahuan struktur dan implementasi

perangkat lunak.

4. Pengujian Jalur

   Pengujian Jalur (*Path Testing*) adalah strategi pengujian struktural yang bertujuan untuk melatih setiap jalur eksekusi independen melalui komponen atau program.

5. Pengujian Integrasi

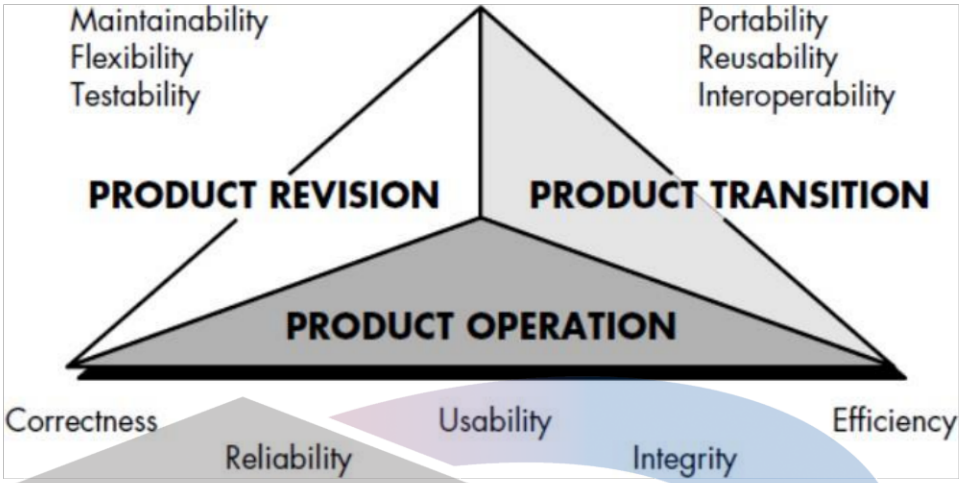
   Kesulitan utama yang muncul pada pengujian integrasi adalah lokalisasi *error* yang ditemukan pada saat proses tersebut. Terdapat interaksi yang rumit antara komponen sistem dan ketika ditemukan *output* yang menyimpang, mungkin sulit untuk menemukan sumber *error* tersebut.

6. Pengujian *Top-down* dan *Bottom-up*

   Pada integrasi *top-down*, komponen sistem tingkat tinggi diintegrasikan dan diuji sebelum perancangan dan implementasinya selesai. Pada integrasi *bottom-up*, komponen tingkat rendah diintegrasikan dan diuji sebelum komponen tingkat yang lebih tinggi dikembangkan. Pengujian *top-down* merupakan bagian integral dari proses pengembangan *top-down* dengan proses pengembangan dimulai dengan komponen tingkat tinggi dan berjalan ke bawah menulusuri hirarki komponen.

2.4 Metode McCall

McCall's Model merupakan model 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 *gap* antara *user* dan *developer*. Dalam membuat *software* yang memiliki performa baik, maka pada saat inisiasi harus menggali kebutuhan dari pengguna secara tepat. Kebutuhan harus didefinisikan secara komprehensif agar menghasilkan *software* yang benar-benar berkualitas [8]. *Software* yang berkualitas akan dihasilkan apabila memperhatikan faktor-faktor kualitas *software* yang terdapat dalam McCall's Model, seperti ditunjukkan pada gambar berikut ini [7].



Gambar 2.3 Faktor Kualitas Perangkat Lunak McCall

Pada dasarnya McCall menitikberatkan faktor-faktor tersebut menjadi 3 (tiga) aspek penting, yaitu yang berhubungan dengan sifat-sifat operasional dari *software* (*Product Operation*), kemampuan *software* dalam menjalani perubahan (*Product Revision*), dan daya adaptasi atau penyesuaian *software* terhadap lingkungan baru (*Product Transition*) [9].

Berdasarkan gambar di atas, penjelasan mengenai faktor-faktor kualitas menurut McCall adalah sebagai berikut [7]:

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.

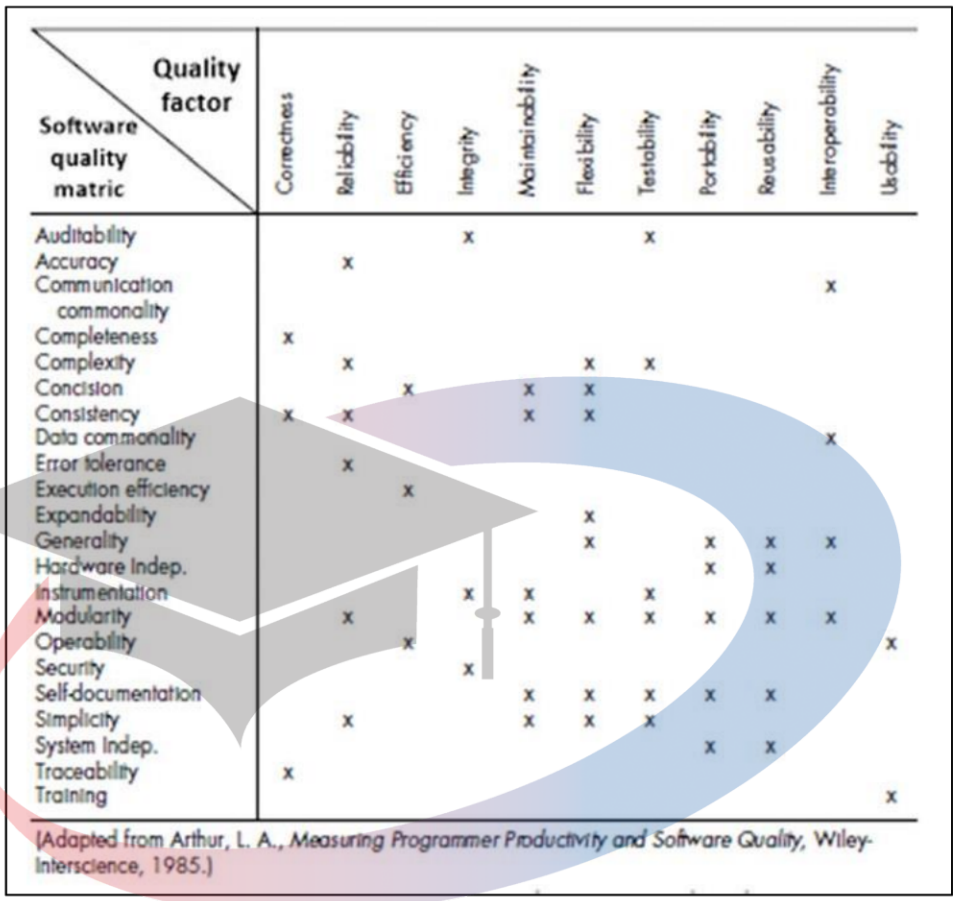
Terdapat beberapa metrik yang digunakan untuk mengukur kuantitas dari kualitas perangkat lunak yang dikembangkan berdasarkan pembagian yang diajukan oleh McCall mengenai formula untuk mengukur faktor-faktor kualitas perangkat lunak. Metrik-metrik berikut ini digunakan dalam pengukuran tersebut [7]:

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 [7].



Gambar 2.4 Hubungan Antara Faktor Kualitas dan Metrik Kualitas Perangkat Lunak

Untuk menghitung kualitas dengan menggunakan metode McCall dapat menggunakan persamaan berikut ini [7].

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

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 [10]:

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.

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