BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem merupakan sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama. Secara garis besar, sebuah sistem informasi terdiri dari tiga komponen utama. Ketiga komponen tersebut mencakup *software, hardware,* dan *brainware*. Ketiga komponen ini saling berkaitan satu sama lain. *Software* mencakup semua perangkat lunak yang dibangun dengan Bahasa pemograman tertentu untuk kemudian menjadi sistem operasi aplikasi, dan driver. *Hardware* mencakup semua perangkat keras yang di satukan menjadi sebuah computer. Dalam konteks yang luas bukan hanya sebuah komputer, namun sebuah jaringan computer. *Brainware* mencakup kemampuan otak manusia, yang mencakup ide, pemikiran, analisis, didalam menciptakan dan mengabungkan *hardware* dan *software*. Dalam pengabungan inilah terciptanya sebuah sistem [5].

Informasi merupakan hasil pengelohan data dari satu atau berbagai sumber, yang kemudian diolah, sehingga memberikan nilai, arti, dan manfaat. Proses ini memerlukan teknologi. Teknologi tersebut tidak harus selalu berkaitan dengan komputer, dengan kata lain, alat tulis dan mesin ketik pun dapat dimasukkan sebagai salah satu teknologi yang digunakan selain kompter dan jaringan komputer [5].

Pada proses pengolahan data, untuk menghasilkan informasi, juga di lakukan proses verifikasi secara akurat, spesifik, dan tepat waktu. Hal ini penting agar informasi dapat memberikan nilai dan pemahaman kepada pengguna. Pengguna dalam hal ini mencakup pembaca, pendengar, penonton, bergantung pada bagaimana cara pengguna tersebut menikmati sajian informasi dan melalui media apa informasi itu di sajikan [5].

Berdasarkan defenisi sistem dan informasi yang telah di jelaskan di atas, maka sistem informasi merupakan gabungan dari empat bagian utama. Keempat bagian utama tersebut mncakup perangkat lunak, perangkat keras,

infrastruktur dan sumber daya manusia. Keempat bagian utama ini saling berkaitan untuk menciptakan sebuah sistem yang dapat mengolah data menjadi informasi yang bermanfaat. Didalamnya juga termasuk poses perencanaan, kontrol, kordinasi, dan pengambilan keputusan. Sehingga sebagai sebuah sistem yang mengolah data menjadi informasi yang akan disajikan da di gunakan pengguna, maka sistem informasi merupakan sebuah sistem yang kompleks, bukan hanya komputer saja yang bekerja, namun juga manusian. Manusia dalam hal ini mengunakan seluruh ide, pemikiran, perhitungan, untuk di tuangkan ke dalam sistem informasi yang digunakan. Dalam sistem informasi terdapat komponen-komponen serta elemen penting didalamnya [5].

Sebuah sistem informasi memiliki sejumlah komponen di dalamnya. Komponen-komponen ini memiliki fungsi dan tugas masing-masing yang saling berkaitan satu sama lain. Keterkaitan antar komponen ini membentuk suatu kesatuan kerja, yang menjadikan sistem informasi dapat mencapai tujuan dan fungsi yang ingin dicapai oleh pengguna dan pengembang sistem informasi [5].

Komponen-komponen pada sistem informasi yaitu:

1. *Input,* mencakup informasi yang masuk, informasi tersebut berupa data yang berasal dari satu maupun sebuah sumber.
2. *Output,* menyajikan hasil akhir ke pengguna sistem informasi. Informasi yang di sajikan sesuai dengan data yang dimasukkan dan fungsionalitas dari sistem informasi bersangkutan.
3. *Software,* mencakup semua perangkat lunak yang digunakam dalam informasi.
4. *Hardware,* mencakup semua perangkat keras computer yang digunakan secara fisik di dalam sistem informasi.
5. *Database,* menyimpan semua data dan informasi ke dalam satu atau beberapa table.
6. Kontrol dan prosedur, mencegah terjadinya berbagai gangguan dan ancaman terhadap data dan informasi yang ada didalam sistem informasi.

7. Teknologi dan jarian Komputer, mengatur *software, hardware, database,* Kontrol dan prosedur, *input, output,* sehingga sistem dapat berjalan dan terkendali dengan baik.

Sistem informasi sebagai sebuah kumpulan lengkap dari perangkat keras, perangkat lunak, basis data, jaringan komputer, pengguna, dan sejumblah prosedur yang telah ter konfigurasi dengan baik, untuk mengumpulkan, mengolah, menyimpan, dan memproses data menjadi informasi [5].

Elemen penting pada sistem informasi yaitu:

1. *Hardware,* mencakup semua perangkat keras komputer yang diperlukan oleh sebuah sistem informasi.
2. *Software,* membantu sistem informasi dalam proses pengoprasian, pengolahan data, pengambilan keputusan, analisis, manajemen data, dan lain-lain.
3. Pengguna, mencakup semuah hirarki kelompok pengguna yang berhubungan dengan sistem informasi,
4. Prosedur, mencakup semua prosedur didalam sistem informasi
5. Basis data, media untuk menyimpan data dan informasi yang dimiliki sistem informasi bersangkutan.
6. Komunikasi, memudahkan pengolahan data, pengguna dan mudah untuk menyajikan informasi ke punguna.

Sistem informasi menjadikan begitu mudah untuk mengakses dan menikmati sajian informasi yang diberikan, sebuah sistem informasi memeberikan banyak manfaat bagi penguna dalam melalukan berbagai hal dimanapun dan kapanpun [5].

Kemudahan dalam menggunakan sistem informasi:

1. Data yang terpusat, sistem informasi menjadikan data dan informasi terkumpul secara terpusat pada suatu server hanya dengan mengaksesnya saja informasi tersebut akan dimunculkan.

2. Kemudahan didalam mengakses informasi, dalam mengakses sistem informasi memberikan kemudahan bagi pengguna untuk mengakses informasi dimanapun dan kapanpun.
3. Efesiensi waktu, dengan adanya sistem informasi, maka kemudahan di dapatkan tanpa harus menghabiskan waktu, cukup dengan terkoneksi ke jaringan maka sistem informasi dapat didapatkan.
4. Cakupan dan penyebaran informasi menjadi lebih luas dan cepat, cakupan informasi yang disajikan tidak lagi hanya untuk perorangan atau beberapa orang saja, namun dapat secara umum ke siapapun yang mengakses sistem informasi.
5. Memudahkan proses bisnis dan pekerjaan, penggunaan sistem informasi, pekerjaan berat dan di lakukan secara manual dapat di kerjakan dengan mudah, otomatis dan lebih hemat waktu dengan hasil yang lebih baik.
6. Biaya murah untuk mengakses dan penyediaan informasi, mengakses informasi tanpa batas hanya dengan membayar biaya internet saja.
7. Menyimpan data lebih banyak dengan ruang yang lebih kecil, penyimpanan digital memudahkan pengguna menyimpan data lebih banyak dengan ruang yang kecil tetapi memiliki resiko data hilang jika sistem informasi tersebut rusak.
8. Solusi komunikasi yang murah, tersediannya sarana komunikasi online membuat pengguna mudah dalam membagi informasi satu sama lain.
9. Penyimpanan data dapat lebih berkembang sesuai kebutuhan, basis data memiliki kemampuan untuk dapat berkembang lebih jauh sesuai kebutuhan.

Peranan sistem infromasi dalam sebuah organisasi adalah untuk meningkatkan, memperluas dan mmperkuat lingkungan manajemen data sesusai dengan organisasi yang bersangkutan. Adapun keperluan dari organisasi ini berdasarkan kepada perencanaan strategis organisasi untuk memprosesan informasi. Sebuah sistem informasi dapat memenuhi kebutuhan akan beragam informasi di dalam organisasi. Pada beberapa kasus, sistem informasi tertentu berperan di dalam pengambilan keputusan (*decision system*).

Jenis-jenis informasi ini memerlukan banyak data untuk dapat turut serta memberikan keputusan yang cepat [5].

Peranan sistem informasi dalam organisasi yaitu:

1. Otomasi pekerja, beragam pekerjaan umumnya dikerjakan sencara manual menjadi otomatis denga mengunakan sistem informasi.
2. *Multi processing,* dapat melakukan pekerjan *input, install, output* secara bersamaan.
3. Pembagian sistem, mencakup kebutuhan pengguna sistem informasi dengan pembagian partisi sistem tersebut.
4. Kombinasi perangkat keras dan perangkat lunak, mencakup proses pekerjaan yang terdapat di organisasi tersebut.
5. Sistem pendukung pengambilan keputusan, membantu level atas organisasi untuk keputusan terkait dengan kebijakan yang ada maupun di berlakukan.

2.2 Rekayasa Perangkat Lunak

Berikut dibawah ini adalah pengertian Rekayasa Perangkat Lunak (RPL) :

1. Perangkat lunak adalah instruksi-instruksi yang ketika di jalankan menyediakan fitur-fitur, fungsi-fungsi, dan kinerja yang di kehendaki, struktur data yang memungkinkan program-program memanipulasi informasi dan informasi deskriptif pada salinan tercetak dan bentuk-bentuk maya yang menggambarkan pengoperasian dan penggunaan program-program [6].
2. Perangkat Lunak adalah instruksi langsung komputer untuk melakukan pekerjaan dan dapat ditemukan di setiap aspek kehidupan modern dari aplikasi yang kritis untuk hidup (*life-critical*), seperti perangkat pemantauan medis dan pembangkit tenaga listrik sampai perangkat hiburan, seperti *video game* [7].
3. Rekayasa perangkat lunak (RPL) adalah aplikasi dari suatu sistem , disiplin, pendekatan dapat dihitung untuk pengembangan, operasi dan pemeliharaan perangkat lunak [7] .

4. Rekayasa perangkat lunak (RPL) adalah produk yang dibangun oleh perangkat lunak profesional dan kemudian mendukung dalam jangka panjang [6].
5. Rekayasa perangkat lunak adalah disiplin rekayasa dengan perangkat lunak yang dikembangkan. Biasanya proses melibatkan penemuan pada keinginan klien, menyusunnya didalam daftar kebutuhan, merancang arsitektur yang mampu mendukung semua kebutuhan, perancangan, pengkodean, pengujianm dan pengintegrasian bagian yang terpisah, menguji keseluruhan, penyebaran dan pemeliharaan perangkat lunak. Sedangkan pemograman hanya menjadi bagian kecil dari rekayasa perangkat lunak [7].

Rekayasa perangkat lunak (RPL) memiliki 4 lapisan [6]. Berikut lapisan-lapisan rekayasa perangkat lunak:

a. *A Quality Focus* : Rekayasa perangkat lunak merupakan teknologi yang berlapis, setiap pendekatan rekayasa harus berkomitmen terhadap kualitas.
b. Proses : merupakan pondasi dari rekayasa perangkat lunak. Proses juga merupakan penghubung lapisan teknologi dan memungkinkan perkembangan perangkat lunak komputer secara rasional dan tepat waktu
c. Metode : menyediakan teknis cara membangun perangkat lunak.
d. Alat : menyediakan otomatis atau semiotomatis dukungan untuk proses dan metode.

2.3 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah proses untuk menemukan kesalahan sebelum di kirim kepada pengguna. Pengujian perangkat lunak merupakan kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean Pentingnya pengujian perangkat lunak adalah untuk dapat menjalankan program dengan maksud mencari kesalahan, dan kasus uji yang baik yaitu kasus yang memiliki peluang untuk mendapatkan kesalahan yang belum diketahui. Pengujian dikatakan berhasil apabila dapat memunculkan kesalahan yang belum diketahui, dan

pengujian yang baik bukan untuk memastikan tidak adanya kesalahan, tetapi untuk mencari sebanyak mungkin kesalahan yang ada pada program [6].

Pengujian perangkat lunak diuji untuk menemukan kesalahan yang dibuat secara tidak sengaja saat perangkat lunak tersebut dirancang dan dibangun. Sebuah strategi untuk pengujian perangkat lunak dikembangkan oleh manajer proyek, rekayasawan perangkat lunak, dan spesiali pengujian. Pengujian sering memerlukan usaha proyek yang lebih disbanding kegiatan rekayasa perangkat lunak yang lain. Jika pengujian dilakukan dengan sembarangan, waktu akan terbuang sia-sia, usaha yang sebenarnya tidak perlu telah terlanjur dikeluarkan, dan bahkan lebih buruk, kesalahan tidak terdeteksi. Karena itu, masuk akal untuk membentuk strategi sistematis untuk pengujian perangkat lunak. Pengujian dimulai "dari yang kecil" dan berlanjut "ke yang besar", pengujian awal berfokus pada komponen tunggal atau sekelompok kecil komponen yang saling terkait dan baru kemudian dilakukan pengujian untuk mengungkap kesalahan dalam data dan logika pemrosesan yang telah tertanam didalam komponen-komponen tersebut. Setelah komponen diperiksa, komponen-komponen itu haruslah diintegrasi sampai sebuah sistem yang lengkap terbangun. Pada titik ini, serangkaian pengujian yang bersifat penting dilaksanakan untuk menemukan kesalahan dalam memenuhi persyaratan pelanggan. Saat kesalahaan ditemukan, kesalahan tersebut harus didiagnosis dan dikoreksi dengan menggunakan sebuah proses yang disebut pelacakan kesalahan (*debugging*). Sebuah spesifikasi pengujian mendokumentasikan pendekatan tim perangkat lunak dalam melakukan pengujian. Caranya dengan mengidentifikasikan sebuah rencana yang menggambarkan keseluruhan strategi dan prosedur yang mendefinisikan langkah-langkah pengujian secara spesifik dan jenis pengujian yang akan dilakukan. Dengan meninjau Spesifikasi Pengujian sebelum melaksanakan pengujian, kita bisa menilai kelengkapan kasus dan tugas pengujian. Perencanaan dan prosedur pengujian yang efektif akan membawa pada konstruksi perangkat lunak yang teratur dan penemuan kesalahan pada setiap tahap dalam proses konstruksi [6].

Sebuah strategi untuk pengujian perangkat lunak harus mengakomodasi pengujian tingkat rendah yang diperlukan untuk memverifikasi bahwa segmen kecil kode sumber telah dilaksanakan dengan benar, sebaik pengujian tingkat tinggi yang memvalidasi fungsi sistem utama terhadap persyaratan pelanggan. Strategi harus memberikan panduan bagi praktisi dan merupakan kumpulan patok kerja (*milestone*) bagi manajer. Karena langkah-langkah dalam strategi pengujian terjadi pada waktu saat tekanan tenggang waktu mulai meningkat, maka kemajuan harus dapat diukur dan masalah harus sedini mungkin muncul ke permukaan [6].

Dari sudut pandang psikologis, analisis dan perancangan perangkat lunak dan kemudian menciptakan program komputer dan dokumentasinya. Dari sudut pandang pembangunan, pengujian dianggap sebagai hal yang merusak. Jadi, lebih mudah bagi pembangun untuk merancang dan menjalankan pengujian yang akan mendemonstrasikan bagaimana program bekerja daripada menemukan kesalahan program tersebut [6].

2.4 Managemen Kualitas Perangkat Lunak

Bahkan pengembang perangkat lunak yang paling letih akan setuju bahwa perangkat lunak berkualitas tinggi adalah tujuan penting. Tetapi bagaimana kita mendefinisikan kualitas? Ada sedikit pertanyaan bahwa definisi sebelumnya dapat dimodifikasi atau diperluas dan diperdebatkan tanpa henti Definisi tersebut berfungsi untuk menekankan tiga poin penting [6]:

1. Persyaratan perangkat lunak adalah dasar dari mana kualitas diukur. Kurangnya kesesuaian dengan persyaratan adalah kurangnya kualitas.
2. Standar yang ditentukan menentukan serangkaian kriteria pengembangan yang memandu cara rekayasa perangkat lunak. Jika kriteria tidak diikuti, kurangnya kualitas hampir pasti akan menghasilkan.
3. Ada serangkaian persyaratan implisit yang sering kali tidak disebutkan (mis., Keinginan untuk kemudahan penggunaan). Jika perangkat lunak sesuai dengan persyaratan eksplisit tetapi gagal memenuhi persyaratan implisit, kualitas perangkat lunak dicurigai.

Kualitas perangkat lunak adalah campuran faktor yang kompleks yang akan bervariasi di berbagai aplikasi dan pelanggan yang memintanya. Pada bagian berikutnya, faktor kualitas perangkat lunak diidentifikasi dan aktivitas manusia yang diperlukan untuk mencapainya dijelaskan [6].

Kebanyakan pengembangan perangkat lunak saat ini akan setuju bahwa perangkat lunak yang berkualitas tinggi merupakan sasaran yang sangat penting. Kualitas perangkat lunak dapat di defenisikan sebagai suatu proses perangkat lunak yang efektif diterapkan dalam arti kata proses perangkat lunak yang menyediakan nilai yang dapat diukur untuk mereka yang menghasilkan dan untuk mereka yang menghasilkannya [6].

Dalam hal ini, ada beberapa pertanyaan kecil dalam konteks defenisi diatas, defenisi tersebut sesungguhnya memberi tekanan lebih pada 3 titik yang penting berikut ini [6].

1. Suatu proses perangkat lunak efektif yang menetapkan infrastruktur yang mendukung setiap usaha untuk mengembangkan produk perangkat lunak yang berkualitas tinggi. Aspek-aspek manajemen proses menciptakan pemeriksaaan dan keseimbangan yang dapat membantu tim perangkat lunak untuk membantu menghapus kan kekacauan pada suatu proyek, yang hal ini sesungguhnya merupakan contributor kunci pada terciptanya kualitas perangkat lunak yang rendah. Praktik-praktik rekayasa perangkat lunak sesungguhnya memungkinkan tim pengembangan perangkat lunak untuk melakukan analisis permasalahan dan merancang suatu solusi atas permasalahan yang baik-keduanya pada dasarnya merupakan hal yang kritis untuk mengembangkan perangkat lunak yang bekualitas tinggi. Terakhir, aktivitas-aktivitas peyangga, seperti pengelolaan perubahan dan tinjauan-tinjauan teknis, sesungguhnya memiliki peran yang penting pada kualitas sebagai suatu bagian dari praktik perangkat lunak.
2. Suatu produk yang bermanfaat memiliki di dalamnya isi, fungsi-fungsi, serta fitur-fitur yang diinginkan oleh para pengguna akhir, namun yang lebih penting, produk tersebut memiliki didalamnya asset-aset yang andal serta bebas dari kesalahan-kesalahan. Produk yang bermanfaat selalu

memenuhi kebutuhan-kebutuhan yang sebelumnya secara eksplisit ditetapkan oleh para stakeholder. Produk tersebut memenuhi pula sejumlah kebutuhan yang bersifat implicit (misalnya kemudahan penggunaan) yang diharapkan ada pada perangkat lunak yang berkualitas tinggi.

3. Dengan melakukan penambahan nilai pada produsen maupun pengguna produk perangkat lunak, perangkat lunak yang berkualitas tinggi selalu memberi keuntungan pada organisasi perangkat lunak dan para pengguna akhir. Organisasi-organisasi pengembangan perangkat lunak akan mendapatkan nilai tambah karena perangkat lunak yang memiliki kualitas tinggi selalu membutuhkan usaha pemeliharaan yang kecil, selalu memiliki kesalahan-kesalahan yang harus diperbaiki dalam jumlah yang kecil, serta selalu mengurangi kebutuhan akan dukungan terhadap pelanggan. Hal ini memungkinkan para rekasayawan perangkat lunak menghabiskan lebih banyak waktu mereka yang sangat berharga untuk membuat aplikasi-aplikasi yang baru serta hanya menyediakan waktu yang lebih sedikit untuk melakukan pekerjaan-pekerjaan ulang yang sesungguhnya perlu diminimalis. Komunitas pengguna akhir pada dasarnya akan mendapatkan nilai tambah karena aplikasi-aplikasi yang dikembangkan oleh organisasi-organisasi perangkat lunak menyediakan kemampuan yang bermanfaat untuk menyelesaikan proses-proses bisnis yang mereka butuhkan.

2.4.1 Faktor-faktor Kualitas yang Menjadi Target

Tim perangkat lunak dapat mengembangkan sejumlah karakteristik kualitas dan menghubungkan pertanyaan-pertanyaan yang akan memperlihatkan faktor-faktor kualitas mana yang telah dipenuhi. Sebagai contoh, McCall mengidentifikasi penggunaan (*usability*) sebagai faktor kualitas yang penting. Jika kita diminta untuk melakukan peninjauan sebuah antarmuka dan menilai ‘penggunaan’ nya, apa yang harus kita lakukan? Kita mungkin bisa mulai dari subatribut-subatribut yang disarankan oleh McCall, kemudian untuk dipahami, kemudian untuk dipelajari, serta kemudian untuk dioperasikan [6]. Untuk memulai penilaian kita terhadap perangkat lunak, kita harus menyelesaikan permasalahan atribut-atribut antarmuka yang bersifat

spesifik atau terukur. Sebagai contoh [6] menyatakan bahwa antarmuka-antarmuka pengguna seharusnya mengikuti faktor-faktor kualitas berikut ini.

a. Intuitif. Derajat bagaimana antarmuka pengguna mengikuti pola-pola penggunaan yang diharapkan sehingga bahkan para pemula dapat menggunakannya tanpa pelatihan yang berlebihan.
b. *Efisiensi*. Derajat bagaimana operasi-operasi dan informasi-informasi dapat ditemukan dan dilaksanakan.
c. Ketangguhan. Berkaitan dengan kemampuan perangkat lunak untuk menangani asupan-asupan (*input*) data yang buruk atau menangani interaksi pengguna yang tidak sesuai.
d. Kekayaan. Derajat tentang bagaimana antarmuka pengguna menyediakan sejumlah fitur yang kaya.

Saat perancangan antarmuka pengguna dikembangkan, tim perangkat lunak seharusnyan melakukan peninjauan langsung terhadap *prototype* rancangan, tim perangkat lunak bisa mengatakan bahwa antarmuka pengguna yang dikembangkan nya sudah memperlihatkan kualitas yang tinggi [6].

2.4.2 Dilema Kualitas Perangkat Lunak

Saat kita berharap dengan dilemma kualitas (pada dasarnya semua orang akan berhadapan dengan hal ini pada suatu waktu tertentu), cobalah untuk mencapai keseimbangan, berusaha sebisa mungkin menghasilkan produk yang kualitasnya dapat diterima, tanpa harus mengorbankan proyek yang menghasilkannya [6].

1. Perangkat Lunak yang “Cukup Bagus”.

   Jika kita menerima argument yang dibuat oleh Meyer, apakah dapat diterima jika kita mengatakan kita harus menghasilkan perangkat lunak yang “cukup bagus”? Jawaban untuk pertanyaan itu seharusnya adalah “ya”, karena sebagian besar perusahaan perangkat lunak melakukan nya setiap hari. Meraka membuat perangkat lunak dengan sejumlah kecil kekurangan (*bug*) dan mengirimkannya ke populasi pengguna yang luas.

   Sesungguhnya apa yang dimaksud dengan “cukup bagus”? perangkat lunak yang cukup bagus sesungguhnya memiliki fungsi-fungsi dan fitur-fitur

berkualitas tinggi sesuai dengan apa yang diinginkan oleh pengguna, tetapi pada saat yang sama memiliki juga fungsi-fungsi serta fitur-fitur terkhususkan yang membuat kesalahan (*bug*) tertentu yang diketahui.

2. Resiko- resiko

   Seharusnya perangkat lunak dikembangkan dengan cara menjadi lebih baik. Implikasi dari perangkat lunak yang berkualitas rendah adalah peningkatan risiko-risiko baik untuk pengembang maupun untuk para pengguna akhir. Pada bagian sebelumnya, telah dibahas tentang risiko-risiko (dalam bentuk biaya) akan tetapi, perancangan yang buruk serta aplikasi yang diimplementasikan tidak selalu berakhir dengan perhitungan-perhitungan yang berkaitan dengan uang dan waktu.

   Kualitas perangkat lunak yang buruk akan memicu risiko-risiko, dan beberapa di antaranya sangat serius.

3. Kualitas dan Keamanan.

   Saat sistem-sistem berbasis web dan aplikasi-aplikasi yang bersifat kritis mulai bertumbuh saat ini, keamanan aplikasi menjadi semakin penting. Dapat dikatakan secara jelas bahwa perangkat lunak- perangkat lunak yang tidak memiliki kualitas yang tinggi akan mudah dibobol oleh para hacker dan sebagai akibatnya, perangkat lunak yang berkualitas rendah secara tidak langsung memiliki risiko keamanan yang tinggi dan memiliki di dalamnya biaya-biaya dan permasalahan-permasalahan tersembunyi.

   Untuk mengembangkan sistem/perangkat lunak yang aman, kita harus selalu berfokus pada kualitas, konsep-konsep dan metode-metode yang terkait dengan arsitektur sistem/perangkat lunak. Dengan mengurangi permasalahan-permasalahan yang berkaitan dengan arsitektur sistem/perangkat lunak (dimana hal ini selanjutnya akan meningkatkan kualitas sistem/perangkat lunak), kita akan mengurangi peluang-peluang terjadinya pembobolan oleh para hacker pada sistem/perangkat lunak yang kita kembangkan.

2.5 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 melatar belakangi model ini adalah karena kurang jelasnya kebutuhan yang ditetapkan untuk mencakup aspek penting dari fungsional sebuah *software* adalah penyebab dari buruknya performa suatu software. Dalam 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* [8].

McCall menitikberatkan faktor-faktor kualitas McCall tersebut menjadi tiga aspek penting, yaitu yang berhubungan dengan [9]:

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 2. 1 Faktor Kualitas Perangkat Lunak McCall

Berdasarkan gambar 2.1, penjelasan mengenai faktor-faktor kualitas menurut McCall sebagai berikut [6]:

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 berhubungan dengan pengemasan dan lingkup fungsi-fungsi yang dilakukan oleh aplikasi/program.

11. Interoperabilitas (*interoperability*)

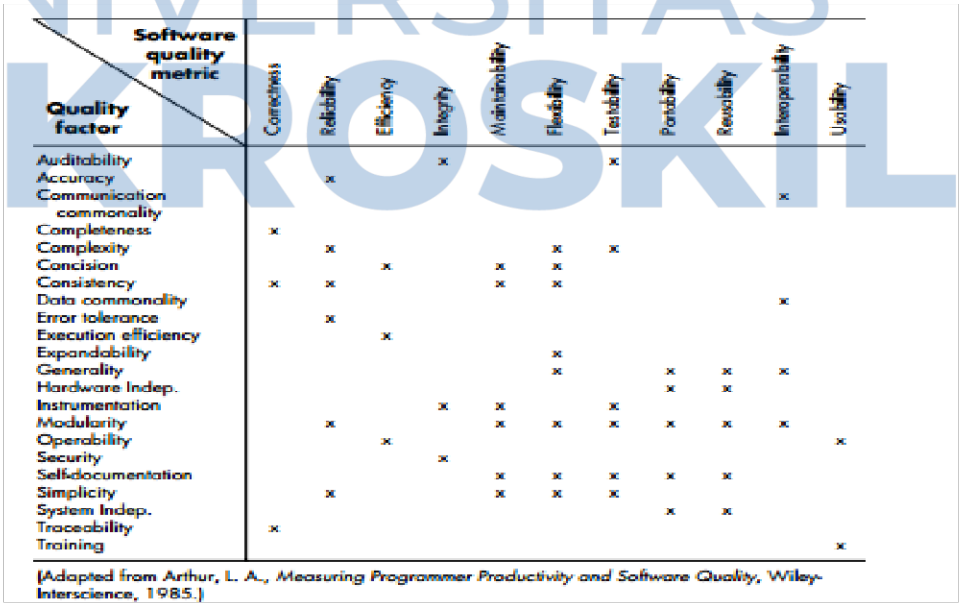
Besarnya usaha yang diperlukan untuk menggantikan bagian suatu sistem dengan bagian sistem yang lainnya.

Terdapat beberapa metrik yang digunakan dalam mengukur kuantitas dari kualitas perangkat lunak yang dikembangkan berdasarkan pembagian yang diajukan oleh McCall mengenai formula untuk mengukur faktor-faktor kualitas perangkat lunak. Metrik berikut yang digunakan dalam pengukuran tersebut yaitu [6]:

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.

Gambar dibawah ini adalah bentuk hubungan antara metrix dengan faktor menggunakan metode McCall.



Gambar 2. 2 Hubungan Metrix dengan Faktor Kualitas McCall

2.6 Teknik Pengukuran McCall

Untuk menghitung kualitas dengan menggunakan metode McCall dapat menggunakan persamaan berikut ini [10].

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

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.

Tabel 2. 1 Kategori Kelayakan

Kategori	Kelas Interval
Sangat Baik	81% - 100%
Baik	61% - 80%
Cukup Baik	41% - 60%
Tidak Baik	21% - 80%
Sangat Tidak Baik	< 21%