BAB II

LANDASAN TEORI

2.1. Kualitas Perangkat Lunak

Kualitas perangkat lunak adalah pemenuhan terhadap kebutuhan fungsional dan kinerja yang didokumentasikan secara eksplisit, pengembangan standar yang didokumentasikan secara eksplisit, dan sifat-sifat implisit yang diharapkan dari sebuah software yang dibangun secara profesional (Dunn, 1990). Berdasarkan definisi di atas terlihat bahwa sebuah perangkat lunak dikatakan berkualitas apabila memenuhi tiga ketentuan pokok:

1. Memenuhi kebutuhan pemakai – yang berarti bahwa jika perangkat lunak tidak dapat memenuhi kebutuhan pengguna perangkat lunak tersebut, maka yang bersangkutan dikatakan tidak atau kurang memiliki kualitas.

2. Memenuhi standar pengembangan perangkat lunak – yang berarti bahwa jika cara pengembangan perangkat lunak tidak mengikuti metodologi standar, maka hampir dapat dipastikan bahwa kualitas yang baik akan sulit atau tidak tercapai; dan

3. Memenuhi sejumlah kriteria implisit – yang berarti bahwa jika salah satu kriteria implisit tersebut tidak dapat dipenuhi, maka

perangkat lunak yang bersangkutan tidak dapat dikatakan memiliki kualitas yang baik.

2.2. SISTEM PENDUKUNG KEPUTUSAN (SPK)

Proses pengambilan keputusan merupakan hal yang menjadi bagian penting di dalam suatu organisasi atau perusahaan. Pengambilan keputusan yang tepat diharapkan dapat meningkatkan kinerja ataupun pengambilan keputusan terhadap kondisi yang ada. Pengambilan keputusan yang tepat juga seharusnya di imbangi dengan kecepatan dan keakuratan dari pengumpulan data, pengolahan data sampai pada akhirnya pada tahap pengambilan keputusan.

1. Definisi Sistem Pendukung Keputusan

   Menurut Keen dan Scoot Morton, Sistem Pendukung Keputusan merupakan penggabungan sumber – sumber kecerdasan individu dengan kemampuan komponen untuk memperbaiki kualitas keputusan. Sistem Pendukung Keputusan juga merupakan sistem informasi berbasis komputer untuk manajemen pengambilan keputusan yang menangani masalah –masalah semi struktur. Sedangkan menurut Turban dkk (2005:140), Sistem Pendukung Keputusan merupakan sistem yang dimaksudkan untuk mendukung para pengambil keputusan manajerial dalam situasi keputusan semi terstruktur. Sistem Pendukung Keputusan dimaksudkan untuk menjadi alat bantu bagi para pengambil keputusan untuk memperluas kapabilitas mereka, namun tidak untuk

menggantikan penilain mereka. Sistem Pendukung Keputusan ditujukan untuk keputusan-keputusan yang memerlukan penilain atau pada keputusan-keputusan yang sama sekali tidak di dukung oleh alogaritma.

2. Komponen Sistem Pendukung Keputusan

Menurut Turban (2005:143), aplikasi sistem pendukung keputusan dibentuk dari subsistem - subsistem, yakni :

a. Subsistem Manajemen Data

Subsistem manajemen data memasukan satu database yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen database (DBMS). Subsistem manajemen data dapat di interkoneksikan dengan data warehouse perusahaan, suatu repositori untuk perusahaan yang relevan untuk pengambilan keputusan. Biasanya data disimpan atau diakses via server web database.

b. Subsistem Manajemen Model

Subsistem manajemen model merupakan paket perangkat lunak memasukan model keuangan, statistik, ilmu manajemen, atau model kuantitatif lainya yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat. Bahasa-bahasa pemodelan untuk membangun model-model kustom yang dimasukan. Perangkat lunak ini sering disebut

sistem manajemen basis model (MBMS). Komponen ini dapat dikoneksikan ke penyimpanan korporat atau eksternal yang ada pada model. Sistem manajemen dan metode solusi model di implementasikan pada sistem pengembangan web untuk berjalan pada server aplikasi.

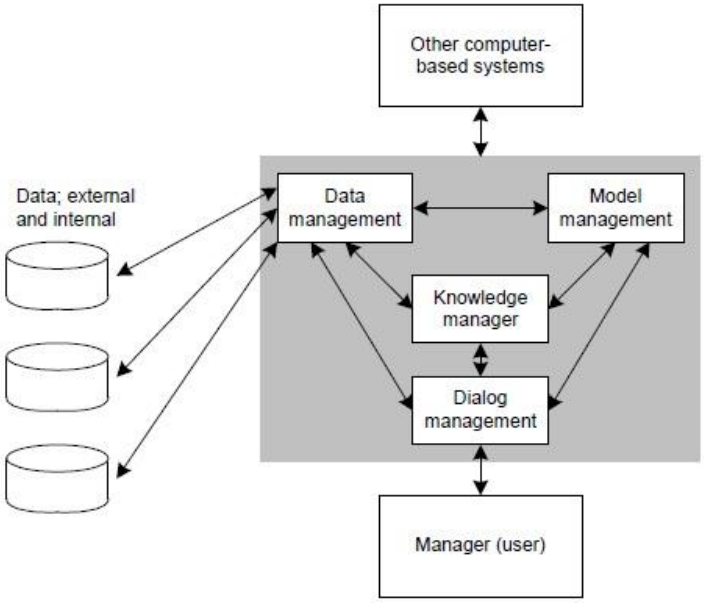
c. Subsistem Manajemen berbasis pengetahuan

Pada subsistem manajemen berbasis pengetahuan, pengguna berkomunikasi dengan dan memerintahkan sistem pendukung keputusan melalui subsistem ini. Pengguna adalah bagian yang dipertimbangkan dari sistem. Para peneliti menegaskan bahwa beberapa kontribusi unik dari sistem pendukung keputusan ini berasal dari interaksi yang intensif antara komputer dan pembuat keputusan. Browser web memberikan struktur antarmuka pengguna grafis yang familiar dan konsisten bagi kebanyakan sistem pendukung keputusan.

d. Subsistem Antarmuka Pengguna

Subsistem ini dapat mendukung semua subsistem lain atau bertindak sebagai suatu komponen independen. Ia memberikan intelegensi untuk memperbesar pengetahuan si pengambil keputusan. Subsistem ini dapat di interkoneksikan dengan repositori pengetahuan perusahaan yang terkadang disebut basis pengetahuan organisasional. Pengetahuan dapat di

sediakan via server web. Banyak metode kecerdasan tiruan di implementasikan dalam sistem pengembangan web seperti java dan mudah untuk di integrasikan dengan komponen sistem pendukung keputusan lainya.



Gambar 2.1. Bagan Skematik Komponen Sistem Pendukung Keputusan

2.3. *TOPSIS (TECHNIQUE FOR OTHERS REFERENCE BY SIMILARITY TO IDEAL SOLUTION)*

TOPSIS adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang pada tahun 1981. *TOPSIS* menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan

jarak Euclidean untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal.

Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut. *TOPSIS* mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan secara praktis. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien,dan memiliki kemampuan mengukur kinerja relatif dari alternatif-alternatif keputusan (Wang dkk, 2006).

2.3.1 Tahapan Perhitungan Metode *TOPSIS*

Langkah-langkah perhitungan dengan menggunakan metode *TOPSIS* (Sri Kusumadewi, 2006) :

a. Membuat matriks keputusan yang ternormalisasi;

b. Membuat matriks keputusan yang ternormalisasi terbobot;

c. Menentukan matriks solusi ideal positif & matriks solusi ideal negatif;

d. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif & matriks solusi ideal negatif;

e. Menentukan nilai preferensi untuk setiap alternatif.

2.3.2 Teknik Perhitungan Metode *TOPSIS*

Menurut Sri Kusumadewi (2006:88), *TOPSIS* membutuhkan rating kinerja setiap alternatif A_i pada setiap kriteria C_j yang ternormalisasi, yaitu:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad ;\backslash \qquad \text{.....} \qquad (2.1)$$

Solusi ideal positif A+ dan solusi ideal negatif A- dapat ditentukan berdasarkan rating bobot ternormalisasi (yij) dengan persamaan sebagai berikut :

$$\begin{aligned} \mathrm{y_{ij}} &= \mathrm{w_i r_{ij}} \\ \mathrm{A^+} &= \left(\mathrm{y_1^+, y_2^+, \cdots, y_n^+}\right); \\ \mathrm{A^-} &= \left(\mathrm{y_1^-, y_2^-, \cdots, y_n^-}\right); \end{aligned} \qquad \text{.....} \qquad (2.2)$$

Dengan

$$y_j^+ = \begin{cases} \max\limits_i y_{ij}; & \textit{jika } \mathrm{j \text{ adalah atribut keuntungan}} \\ \min\limits_i y_{ij}; & \mathrm{jika\ j\ adalah\ atribut\ biaya} \end{cases}$$

$$y_j^- = \begin{cases} \min\limits_i y_{ij}; & \textit{jika } \mathrm{j \text{ adalah atribut keuntungan}} \\ \max\limits_i y_{ij}; & \mathrm{jika\ j\ adalah\ atribut\ biaya} \end{cases}$$

Jarak antara alternatif *Ai* dengan solusi ideal positif dirumuskan dengan persamaan sebagai berikut:

$$D_i^+ = \sqrt{\sum_{j=1}^{n}\left(y_i^+ - y_{ij}\right)^2}; \qquad \text{.....} \qquad (2.3)$$

Jarak antara alternatif *Ai* dengan solusi ideal negatif dirumuskan dengan persamaan sebagai berikut :

$$D_i^- = \sqrt{\sum_{j=1}^{n}\left(y_{ij} - y_i^-\right)^2}; \qquad \text{.....} \qquad (2.4)$$

Nilai preferensi untuk setiap alternatif (*Vi*) diberikan persamaan sebagai berikut :

$$V_i = \frac{D_i^-}{D_i^- + D_i^+}; \qquad \text{.....} \qquad (2.5)$$

Nilai *Vi* yang lebih besar menunjukkan bahwa alternatif *Ai* lebih dipilih.

2.4. CSS *(Cascading Style Sheets)*

Style sheet merupakan template yang mengontrol pemformatan *tag HTML* pada sebuah halaman web. Konsep *style sheet* mirip dengan template pada Microsoft Word. Dengan *CSS, web designer* dapat mengubah tampilan halaman *web* dengan mengubah format pada *tag HTML* tertentu melalui *style sheet,* untuk selanjutnya mengganti spesifikasi *default* dari *browser* untuk *tag-tag* tersebut. *CSS* digunakan para *web design* untuk mengatur *sytle* elemen yang ada dalam halaman

web, mulai dari memformat teks, sampai pada memformat *layout.* Tujuan dari penggunaan *CSS* ini adalah agar di peroleh suatu konsistensi *style* pada element tertentu. Sebagai contoh, misalnya untuk mengatur *style* elemen *heading*, di inginkan jenis *font* Arial, ukuran 20 *pixel*, serta berwarna merah. Dengan CSS, cukup menuliskan properti dari elemen *heading* tersebut sekali saja.

Terdapat beberapa cara yang dapat dilakukan untuk menambahkan *style sheet* pada sebuah halaman *web*, diantaranya adalah sebagai berikut

a. Dengan membuat *link* ke file *style sheet* dari file *HTML*. Metode ini memungkinan web designer mengubah *style* sejumlah halaman *web* hanya dengan mengedit satu file *style sheet.*

b. Dengan menyisipkan *style sheet pada* file *HTML*. Metode ini memungkinkan *web desginer* mengubah *style* suatu halaman *web* hanya dengan mengedit beberapa baris *style sheet.*

c. Dengan menyisipkan secara *inline* pada *tag* dalam file *HTML*. Hal ini memberi cara tercepat untuk mengubah suatu *tag*, sejumlah *tag*, atau satu blok informasi pada halaman *web.*

Baik *style sheet* yang di *link* maupun yang di sisipkan, memiliki satu atau lebih definisi *style.* Untuk sintaks *inline* mungkin berbeda. Suatu definisi *style* terdiri dari suatu *tag HTML*, di ikuti, oleh sejumlah properti untuk *tag* tersebut yang terletak di antara tanda kurung kurawal. Tiap

properti di identifikasi oleh nama properti, di ikuti oleh titik dua dan nilai properti. Properti ganda di pisahkan oleh titik koma, sebagai contoh , definisi *style* berikut memberi *tag* <H1> ukuran *font* yang spesifik (15 *point),* dan ketebalan huruf (*boldface).*

```
H1 { font-size : 15pt; font-weight : bold; }
```

2.5. *PHP (Hypertext Preprocessor)*

Menurut Didik Dwi Prasetyo (2004:76) *PHP* merupakan bahasa scripting *server-side*, dimana pengolahan datanya dilakukan pada sisi server. Lebih sederhananya, sisi *server* yang akan menerjemahkan skrip atau sintak program, kemudian hasil terjemahan tersebut dikirim kepada *client* yang melakukan permintaan. Seluruh aplikasi berbasis web dapat dibuat dengan *PHP*, namun kelebihan utama *PHP* adalah pada konektivitasnya dengan sistem database di dalam web. Beberapa kelebihan dari *PHP* diantaranya adalah :

a. *PHP* mudah untuk dibuat dan dijalankan, artinya *PHP* dapat berjalan dalam Web Server dan dalam Sistem Operasi yang berbeda pula.

b. *PHP* merupakan perangkat lunak open-source yang gratis dan bebas didistribusikan kembali di bawah lisensi *GPL (GNU Public License).*

c. Banyaknya Web Server yang mendukung *PHP*, seperti *Apache, PWS, IIS*, dan lain-lain.

d. *PHP* didukung oleh banyak database, semisal *MySQL, PostgreSQL, Interbase, SQL*, dan lain-lain.

e. Sintak bahasa pemograman *PHP* lebih sederhana, singkat dan mudah untuk dipahami.

f. *HTML-embedded*, artinya *PHP* adalah bahasa yang dapat ditulis dengan menempelkan pada sintak-sintak *HTML*.

2.6. *MySQL*

Menurut Didik Dwi Prasetyo (2004:18), *MySQL* merupakan salah satu *database server* yang berkembang di lingkungan *open source* dan didistribusikan secara *free* (gratis) dibawah lisensi *GPL*. *MySQL* merupakan *RDBMS (Relational Database Management System)* server. *RDBMS* adalah program yang memungkinkan pengguna database untuk membuat, mengelola, dan menggunakan data pada suatu model relational. Dengan demikian, tabel-tabel yang ada pada *database* memiliki relasi antara satu tabel dengan tabel lainnya. Beberapa keunggulan dari *MySQL* yaitu :

a. Cepat, handal dan mudah dalam penggunaannya

MySQL lebih cepat tiga sampai empat kali dari pada database server komersial yang beredar saat ini, mudah diatur dan tidak memerlukan seseorang yang ahli untuk mengatur administrasi pemasangan *MySQL*.

b. Didukung oleh berbagai bahasa

Database server *MySQL* dapat memberikan pesan error dalam berbagai bahasa seperti Belanda, Portugis, Spanyol, Inggris, Perancis, Jerman, dan Italia.

c. Mampu membuat tabel berukuran sangat besar

Ukuran maksimal dari setiap tabel yang dapat dibuat dengan *MySQL* adalah 4 GB sampai dengan ukuran file yang dapat ditangani oleh sistem operasi yang dipakai.

d. Lebih Murah

MySQL bersifat open source dan didistribusikan dengan gratis tanpa biaya untuk *UNIX* platform, OS/2 dan *Windows* platform.

e. Melekatnya integrasi *PHP* dengan *MySQL*

Keterikatan antara *PHP* dengan *MySQL* yang sama-sama perangkat lunak opensource sangat kuat, sehingga koneksi yang terjadi lebih cepat jika dibandingkan dengan menggunakan database server lainnya. Modul MySQL di *PHP* telah dibuat built-in sehingga tidak memerlukan konfigurasi tambahan pada file konfigurasi *php*.ini.

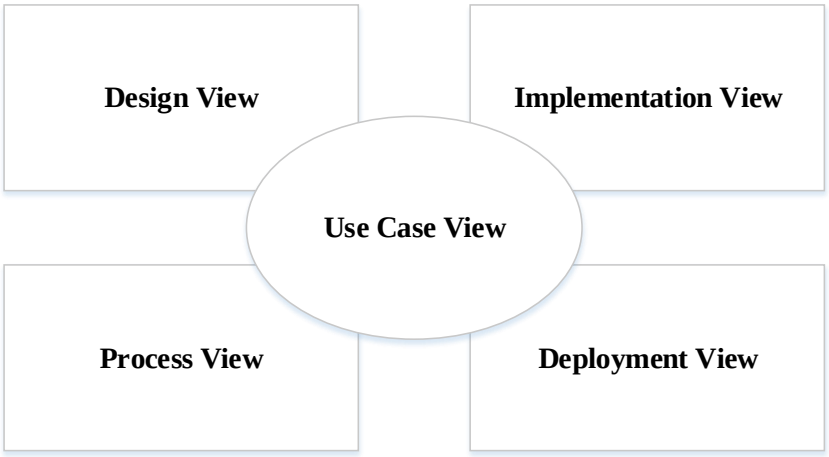
2.7. *UML (Unified Modeling Language)*

Menurut Adi Nugroho (2005:06) *UML* atau *Unified Modeling Language* adalah sebuah bahasa pemodelan untuk sistem atau perangkat lunak yang berorientasi objek. Pemodelan sesungguhnya digunakan untuk penyederhanan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Menurut Munawar (2005), dengan *UML*, akan bisa menceritakan apa yang seharusnya dilakukan oleh sebuah sistem bukan bagaimana yang seharusnya dilakukan oleh sebuah sistem.

Tiga karakter penting yang melekat dalam *UML*, yakni sketsa, cetak biru,bahasa pemrograman. Sebagai sebuah sketsa, *UML* bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek sistem, dengan demikian, semua anggota akan mempunyai gambaran yang sama tentang suatu sistem. *UML* juga bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka bisa diketahui informasi detil tentang koding program (*forward engineering*) atau bahkan membaca program dan mengintrepretasikanya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokumentasi akan di modifikasi/dipelihara. Hal ini bisa terjadi ketika dokumentasi asli hilang atau bahkan belum dibuat sama sekali. Sebagai bahasa pemrograman,

UML dapat menterjemahkan diagram yang ada di *UML* menjadi kode program yang siap untuk dijalankan.



Gambar 2.2 Bagan Model 4+1 View *Unified Modelling Language*

UML dibanguin atas model 4+1 *view*. Model ini didasarkan pada fakta bahwa struktur sebuah sistem di deskripsikan dalam 5 *view* dimana salah satu diantaranya *use case view*. *Use case view* ini memegang peranan khusus untuk mengintegrasikan content ke *view* yang lain.

Kelima *view* tersebut tidak berhubungan dengan diagram yang di deskripsikan di *UML*. Setiap v*iew* berhubungan dengan prespektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili ketertarikan sekelompok *stakeholder* tertentu. Penjelasan lengkap tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

Use case view mendefinisakn perilaku eksternal sistem. Hal ini menjadi daya tarik bagi end user, analis dan tester. Pandangan ini

mendefinisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari rancangan sistem. Itulah sebabnya *use case view* menjadi pusat peran dan sering dikatakan yang mendrive proses pengembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan *use case*. *Design view* ini berisi definisi komponen program, clas-clas utama bersama-sama dengan spesifikasi data, perilaku dan interaksinya. Informasi yang terkandung di view ini menjadi perhatian para programer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan di implementasikan.

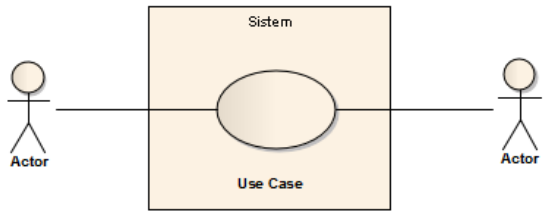
Implementation view menjelaskan komponen-komponen fisik dari sistem yang akan dibangun. Hal ini berbeda dengan komponen logic yang akan dideskripsikan pada design view. Termasuk disini di antaranya file.exe, library dan database. Informasi yang ada di view ini relevan dengan aktifitas-aktifitas seperti manajemen konfigurasi dan integrasi sistem.

Process view berhubungan dengan hal-hal yang berkaitan dengan dengan *concurrency* di dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua view ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yag berhubugan dengan kinerja.

a. *Use Case* Diagram

Use case menggambarkan fungsi-fungsi sistem dari sudut pandang pengguna eksternal dan dalam sebuah cara yang mudah dipahami. *Use case* merupakan penyusunan kembali lingkup fungsional sistem yang disederhanakan lagi. *Use Case* diagram adalah diagram yang menggambarkan interaksi antara sistem dengan sistem eksternal pengguna.

Menurut Munawar (2005), *use case* merupakan alat bantu terbaik dalam menstimulasi pengguna potensial untuk mengatakan tentang suatu system dari sudut pandangnya. Ide dasarnya adalah bagaimana melibatkan penggunaan sistem difase-fase awal analis dan perancangan sistem. Dengan demikian diharapkan akan bisa dibangun suatu sistem yang bisa membantu pengguna. Dalam use case diagram menunjukan tiga aspek dari sistem yakni : *actor*, *use case*, dan sistem/sub sistem boundary. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case.*



Gambar 2.3. Use Case Model *Unified Modeling Language*

b. Class Diagram

Menurut Indrajani (2010:35), *Class* diagram biasanya digunakan untuk menggambarkan perbedaan yang mendasar antara class-class, hubungan antara class, dan di mana sub-sistem *class* tersebut. *Class* diagram bersifat dinamis dikarenakan memperlihatkan instansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas. Diagram ini bersifat statis dikarenakan memperlihatkan himpunan-himpunan kelas-kelas, antar muka-antar muka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umumnya dijumpai pada pemodelan sistem berorientasi objek, meskipun bersifat statis, sering pula diagram ini memuat kelas-kelas aktif. (Adi Nugroho. 20010:30)

c. *Sequence* Diagram

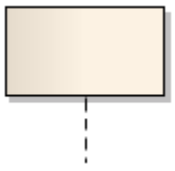
Menurut Indrajani (2010:36), *Sequence* diagram merupakan suatu diagram interaksi yang menggambarkan bagaimana objek-objek berpartisipasi dalam bagian interaksi dan pesan yang ditukar dalam urutan waktu. *Sequence* diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang di hasilkan.

Sequence diagram hanyalah sebagai visualisasi bagaimana obyek berinteraksi daripada sebagai cara untuk pemodelan logika. Didalam *sequence* diagram terdapat tiga point utama yakni : *Obyek/Participant, Message,* serta *Time*

1) Obyek /*Participant*

Setiap *participant* terhubung dengan garis titik-titik yamng disebut *lifeline.* Sepanjang *lifeline* terdapat kotak yang disebut *activation. Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.*



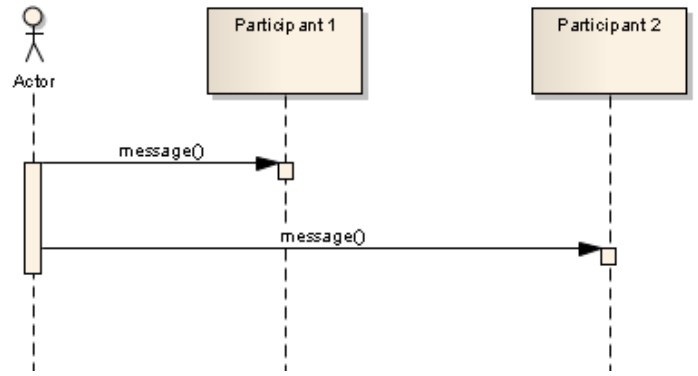
Gambar 2.4. Participant pada sebuah *sequence* diagram.

2) Message

Sebuah *message* bergerak dari satu participant ke participant lain dan dari satu *lifeline* ke *lifeline* lain. Sebuah participant bisa mengirim sebuah message kepada dirinya sendiri. Sebuah message bisa menjadi simple, *synchronous,* atau *asynchronous.* *Message* yang simple adalah sebuah perpindahan kontrol dari satu participant ke participant yang lainya. Jika sebuah participant mengirimkan sebuah *message synchronous,* maka jawaban atas *message* tersebut akan ditunggu sebelum diproses dengan urusannya. Namun jika *message asynchronous* yang dikirimkan, maka jawabn atas *message* tersebut tidak perlu ditunggu.

3) Time

Time merupakan diagram yang mewakili waktu pada arah vertical. Waktu di mulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibandingkan *message* yang lebih dekat ke bawah.



Gambar 2.5. Simbol-simbol pada *sequence* diagram

d. *Activity* Diagram

Activity diagram merupakan teknik untuk mendiskripsikan logika prosedural, proses bisnis, dan aliran kerja dalam banyak kasus. *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. (Munawar.2005:109). *Activity* diagram menunjukan tahapan, pengambilan keputusan, dan percabangan. Diagram ini sangat berguna untuk menunjukan operation sebuah obyek dan proses bisnis. Kelebihan *activity* diagram dibandingkan *flowchart* adalah kemampuannya dalam menampilkan aktivitas paralel.

Activity diagram diperlukan untuk menggambarkan apa yang terjadi ketika aksi atau aktifitas dari suatu state diselesaikan, ketika aliran kendali akan menuju ke aksi berikutnya atau aktifitas.

a. Simbologi

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork
	Rake; menunjukan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Tanda akhir (Flow Final)

Gambar 2.6. Simbologi dalam *activity* diagram.

2.8. *SUPPLIER*

Supplier merupakan suatu perusahaan atau individu yang menyediakan sumber daya yang dibutuhkan oleh perusahaan dan para pesaing untuk memproduksi barang dan jasa tertentu. Suatu perusahaan akan mencari *supplier* yang mutu dan efisiensinya dapat dipertahankan. Hal ini dikarenakan perkembangan dalam *supplier* itu sendiri dapat memberikan pegaruh yang sangat penting terhadap pelaksanaan pemasaran suatu perusahaan. Pada hakikatnya, pemilihan *supplier* dalam rangka rantai *supply* tidak jauh berbeda dengan memilih kebutuhan perusahaan untuk di beli. Perbedaan yang utama hanya terletak pada *supplier* yang memiliki kedudukan yang jauh lebih penting. Oleh karena itu penelitian

dan pertimbangan harus lebih lengkap dan menyeluruh, meskipun tahapan penentuan *supplier* dapat dilakukan dengan beberapa tahapan, dimana perusahaan meninjau, mengevaluasi, serta memilih *supplier* nya untuk menjadi bagian dari rantai *supply* perusahaan.

2.8.1 Kriteria Supplier

Suatu perusahaan atau organisasi membutuhkan *supplier* yang memahami apa yang diharapkan (tujuan) serta siapa yang telah diberi tanggapan atas kinerja *supplier* (umpan balik). Komunikasi ini membantu ke arah menyamakan usaha dalam setiap organisasi atau perusahaan dan dapat merangsang aktivitas sehingga meningkatkan kinerjanya. Beberapa kriteria dari *supplier* yang menjadi bahan pertimbangan bagi perusahaan di dalam proses pemilihan *supplier*:

a. Harga penawaran (*Quotaition Price*)
b. Keandalan dalam ketepatan waktu
c. Frekuensi penyerahan
d. Jumlah pengiriman minimum
e. Mutu *supplier*
f. Biaya angkutan
g. Penyerahan pembayaran
h. Pajak dan nilai tukar

2.9. *McCall*

McCall dan kawan-kawan pada tahun 1977 telah mengusulkan suatu penggolongan faktor-faktor atau kriteria yang mempengaruhi kualitas perangkat lunak (Triyanto dan Charolina, Astri. 2011). Pada dasarnya, McCall menitikberatkan faktor-faktor tersebut menjadi tiga aspek penting, yaitu yang berhubungan dengan:

1. Sifat-sifat operasional dari perangkat lunak *(Product Operations).*
2. Kemampuan perangkat lunak dalam menjalani perubahan *(Product Revision).*
3. Daya adaptasi atau penyesuaian perangkat lunak terhadap lingkungan baru *(Product Transition).(* Dunn R.H , 1990)

1. Product Operations

Sifat-sifat operasional suatu perangkat lunak berkaitan dengan hal-hal yang harus diperhatikan oleh para perancang dan pengembang yang secara teknis melakukan 5 penciptaan sebuah aplikasi. Hal-hal yang diukur di sini adalah yang berhubungan dengan teknis analisa, perancangan, dan konstruksi sebuah perangkat lunak (Triyanto dan Charolina, Astri. 2011). Faktor-faktor McCall yang berkaitan dengan sifat-sifat operasional perangkat lunak adalah:

a. Correctness adalah sejauh mana suatu perangkat lunak memenuhi spesifikasi dan *mission objective* dari users.

b. *Reliability* adalah sejauh mana suatu perangkat lunak dapat diharapkan untuk melaksanakan fungsinya dengan ketelitian yang diperlukan.

c. *Efficiency* adalah banyaknya sumber daya komputasi dan kode program yang dibutuhkan suatu perangkat lunak untuk melakukan fungsinya.

d. *Integrity* adalah sejauh mana akses ke perangkat lunak dan data oleh pihak yang tidak berhak dapat dikendalikan dan

e. *Usability* adalah usaha yang diperlukan untuk mempelajari, mengoperasikan, menyiapkan input, dan mengartikan output dari perangkat lunak.

2. *Product Revision*

Setelah sebuah perangkat lunak berhasil dikembangkan dan diimplementasikan, akan terdapat berbagai hal yang perlu diperbaiki berdasarkan hasil uji coba maupun evaluasi. Sebuah perangkat lunak yang dirancang dan dikembangkan dengan baik, akan dengan mudah dapat direvisi jika diperlukan. Seberapa jauh perangkat lunak tersebut dapat diperbaiki merupakan faktor lain yang harus diperhatikan (Triyanto dan Charolina, Astri. 2011). Faktor-faktor McCall yang berkaitan dengan kemampuan perangkat lunak untuk menjalani perubahan adalah:

a. *Maintainability* adalah usaha yang diperlukan untuk menemukan dan memperbaiki kesalahan (error) dalam perangkat lunak.

b. *Flexibility* adalah usaha yang diperlukan untuk melakukan modifikasi terhadap perangkat lunak yang operasional.

c. *Testability* adalah usaha yang diperlukan untuk menguji suatu perangkat lunak untuk memastikan apakah melakukan fungsi yang dikehendaki atau tidak. (Dunn R.H , 1990).

3. *Product Transition*

Setelah integritas perangkat lunak secara teknis telah diukur dengan menggunakan faktor produk operational dan secara implementasi telah disesuaikan dengan faktor *product revision*, faktor terakhir yang harus diperhatikan adalah faktor transisi yaitu bagaimana perangkat lunak tersebut dapat dijalankan pada beberapa platform atau kerangka sistem yang beragam (Triyanto dan Charolina, Astri. 2011). Faktor-faktor *McCall* yang berkaitan dengan tingkat adaptabilitas perangkat lunak terhadap lingkungan baru:

a. *Portability* adalah usaha yang diperlukan untuk mentransfer perangkat lunak dari suat hardware dan/atau sistem perangkat lunak tertentu agar dapat berfungsi pada hardware dan/atau sistem perangkat lunak lainnya.

b. *Reusability* adalah sejauh mana suatu perangkat lunak (atau bagian perangkat lunak) dapat dipergunakan ulang pada aplikasi lainnya.

c. *Interoperability* adalah usaha yang diperlukan untuk menghubungkan satu perangkat lunak dengan lainnya. (Dunn R.H, 1990).

Pengukuran mengenai faktor-faktor kualitas perangkat lunak dapat dilakukan dengan melakukan penilaian dan pengukuran secara objektif seperti. Pengukuran biasanya menggunakan skala 0-10 dalam proses penilaian. *McCall* menetapkan beberapa pengukuran yang dapat digunakan, diantaranya:

a. *Auditability* adalah kemudahan untuk memeriksa apakah perangkat lunak memenuhi standard atau tidak.

b. *Accuracy* adalah ketelitian dari komputasi dan kontrol

c. *Communication Commonality* adalah sejauh mana interface, protokol, dan bandwidth digunakan.

d. *Completeness* adalah sejauh mana implementasi penuh dari fungsi-fungsi yang diperlukan telah tercapai.

e. *Conciseness* adalah keringkasan program dalam ukuran *LOC (line of commands*).

f. *Consistency* adalah derajat penggunaan teknik-teknik desain dan dokumentasi yang seragam pada seluruh proyek pengembangan perangkat lunak.

g. *Data Commonality* adalah derajat penggunaan tipe dan struktur data baku pada seluruh program.

h. *Error Tolerance* adalah kerusakan yang terjadi apabila program mengalami *error.*

i. *Execution Efficiency* adalah kinerja *run-time* dari program.

j. *Expandability* adalah sejauh mana desain prosedur, data, atau arsitektur dapat diperluas.

k. Generality adalah luasnya kemungkinan aplikasi dari komponen-komponen program.

l. *Hardware Independence* adalah sejauh mana perangkat lunak tidak bergantung pada kekhususan dari *hardware* tempat perangkat lunak itu beroperasi.

m. *Instrumentation* adalah sejauh mana program memonitor operasi dirinya sendiri dan mengidentifikasi error yang terjadi.

n. *Modularity* adalah functional *independence* dari komponen-komponen program.

o. *Operability* adalah kemudahan mengoperasikan program.

p. *Security* adalah ketersediaan mekanisme untuk mengontrol dan melindungi program dan data terhadap akses dari pihak yang tidak berhak.

q. *Self-Documentation* adalah sejauh mana *source-code* memberikan dokumentasi yang berarti.

r. *Simplicity* adalah kemudahan suatu program untuk dimengerti.

s. *Traceability* adalah kemudahan merujuk balik implementasi atau komponen program ke kebutuhan pengguna perangkat lunak.

t. *Training* adalah sejauh mana perangkat lunak membantu pemakaian baru untuk menggunakan sistem

Tabel 2.1 Tabel Faktor Kualitas Berdasarkan Metode McCall

No	Faktor	Kriteria Kualitas
1.	Ketepatan *(Correctness)*	Kelengkapan, konsistensi, ketelusuran
2.	Keandalan *(Reliability)*	Akurasi, toleransi kesalahan, konsistensi, kesederhanaan
3.	Efisiensi *(Eficiency)*	Efisiensi eksekusi, efisiensi penyimpanan
4.	Integritas *(Integrity)*	Kontrol akses, akses audit
5.	Kegunaan *(Usability)*	Komunikasi, pengoperasian, training
6.	Perbaikan *(Maintainability)*	Konsistensi, sederhana, singkat, teratur, dokumentasi diri
7.	Pengetesan *(Testability)*	Kesederhanaan, teratur, instrumentasi, dokumentasi diri
8.	Fleksibilitas *(Flexibility)*	Upgrade, umum, modularitas, dokumentasi diri

9.	Portabilitas *(Portability)*	Sistem kebebasan *software*, kebebasan *hardware*, dokumentasi diri, modularitas
10.	Penggunaan kembali *(Reusability)*	Umum, sistem kebebasan *software*, kebebasan *hardware*, dokumentasi diri, modularitas
11.	Interoperabilitas *(Interoperability)*	Komunikasi, *commonality*, *commonality data*, modularitas.

Sumber : Avin dkk, 2014.

Dari sebelas faktor kualitas menurut taksonomi *McCall* seperti ditunjukkan pada Tabel 2.1, untuk menentukan kualitas perangkat lunak cukup dengan lima faktor. Lima faktor untuk menentukan kualitas perangkat lunak tersebut adalah faktor Ketepatan *(Correctness),* Keandalan *(Reliability),* Efisiensi *(Efficiency),* Kegunaan *(Usability),* dan Perbaikan *(Maintainability)* (Romi Satria Wahono, 2006)

2.9.1 Teknik Perhitungan Metode *McCall*

Untuk mengukur kualitas suatu perangkat lunak dengan menggunakan metode *McCall*, maka dibutuhkan rumus perhitungan sebagai berikut (Romi Satrio W. 2006) :

$$Fa = w1c1 + w2c2 + w3c3 + ... + wncn \quad \text{.....} \quad (2.6)$$

dimana,

Fa = Nilai total dari faktor a

w = Bobot yang bergantung pada produk dan kepentingan kriteria

c = Metrik yang mempengaruhi faktor kualitas perangkat lunak

Dari rumus tersebut, terdapat beberapa tahap dalam penghitungan. Adapun tahapan penghitungan sebagai berikut :

1. Tentukan kriteria yang digunakan
2. Tentukan bobot (w) dari setiap kriteria ($0 \leq w \leq 1$)
3. Tentukan skala dari nilai setiap kriteria
4. Berikan nilai pada tiap kriteria
5. Hitung nilai total Fa (Romi Satrio W, 2006).