BAB II
TINJAUAN PUSTAKA

2.1 Teori Yang Berhubungan Dengan Sistem Secara Umum

Teori yang berhubungan dengan sistem meliputi sistem dan data.

2.1.1 Sistem

Menurut Pratama (2014:7), Sistem dapat didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama.

Sedangkan menurut Pratiwi (2016:4), Sistem adalah kumpulan dari obyek-obyek seperti orang, *resources*, konsep dan prosedur yang ditujukan untuk melakukan fungsi tertentu atau memenuhi suatu tujuan. Kemudian sistem juga merupakan kumpulan dari komponen yang berinteraksi bersama-sama secara kolektif untuk melaksanakan tugas.

Berdasarkan uraian tersebut maka dapat disimpulkan bahwa pengertian sistem adalah sekumpulan prosedur atau diartikan sebagai sekumpulan obyek-obyek yang saling terhubung satu dengan yang lain, yang berinteraksi bersama-sama untuk mencapai tujuan.

2.1.2 Data

Menurut Husda dan Wangdra (2017:13), Data merupakan sesuatu yang belum mempunyai arti bagi penerimanya dan masih memerlukan adanya suatu pengolahan. Data bisa berujut suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa ataupun symbol-simbol lainnya yang bisa kita gunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep.

Sedangkan menurut Sutabri (2012:1), Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata.

Berdasarkan pengertian di atas dapat disimpulkan bahwa data merupakan catatan yang fakta dalam bentuk, angka, huruf, simbol-simbol maupun gagasan yang dapat menggambarkan suatu kejadian yang nyata.

2.2 Teori yang Berhubungan dengan Penelitian

2.2.1 Keputusan

Menurut Pratiwi (2016:2), Keputusan merupakan hasil pemikiran berupa pemilihan satu diantara beberapa alternatif yang dapat digunakan untuk memecahkan masalah yang dihadapi.

Jenis-jenis keputusan dibedakan menjadi tiga macam yaitu keputusan terstruktur, keputusan tidak terstruktur, dan keputusan semi terstruktur (Pratiwi, 2016:5-6).

1. Keputusan terstruktur

   Keputusan-keputusan yang berkaitan dengan persoalan yang telah diketahui sebelumnya. Proses pengambilan keputusan seperti ini biasanya didasarkan atas teknik-teknik tertentu dan sudah dibuat standarnya. Kategori keputusan ini juga dapat dikatakan suatu proses jawaban secara otomatis pada kebijakan yang sudah ditentukan sebelumnya. Secara alamiah hampir semua masalah rutin dan berulang memiliki parameter-parameter persoalan yang telah diketahui dan terdefinisi dengan baik, sehingga jawaban atau proses pengambilan keputusan pun bersifat rutin dan terjadwal.

2. Keputusan tak terstruktur

   Keputusan-keputusan yang berkaitan dengan berbagai persoalan baru. Keputusan tidak terstruktur biasanya juga berkaitan dengan persoalan yang cukup pelik, karena banyak parameter yang tidak diketahui atau belum diketahui. Oleh karena itu, untuk mengambil keputusan ini biasanya intuisi serta pengalaman seorang pelaku organisasi akan sangat membantu.

   Keputusan tak terstruktur, adalah "*fuzzy*", permasalahan kompleks dimana tak ada solusi yang mengikutinya. Masalah yang tak terstruktur adalah tak adanya 3 fase proses yang terstruktur. Keputusan tidak terstruktur (unstructured decision) bukan merupakan keputusan yang berulang dan rutin. Contohnya adalah memilih sampul depan sebuah majalah, mengontrak manajemen tingkat senior, dan memilih proyek penelitian awal yang akan dilakukan. Tidak ada kerangka atau model yang dapat memecahkan masalah sejenis ini. Bahkan, dibutuhkan banyak sekali pertimbangan dan intuisi. Walaupun demikian, keputusan tidak terstruktur dapat didukung oleh bantuan dari keputusan yang diambil berdasar hasil komputer, yang berfungsi untuk memfasilitasi

pengumpulan informasi dari berbagai sumber. Contohnya adalah keputusan untuk pengembangan teknologi baru, pengembangan jenis usaha baru, keputusan untuk bergabung dengan perusahaan lain, perekrutan eksekutif.

3. Keputusan semi terstruktur

   Terdapat beberapa keputusan terstruktur, tetapi tak semua dari fase-fase yang ada. Keputusan semi terstruktur (*semistructured decision*) ditandai dengan dengan peraturan-peraturan yang tidak lengkap untuk mengambil keputusan, dan adanya kebutuhan untuk membuat penilaian serta petimbangan subjektif sebagai pelengkap analisis data yang formal. Menetapkan anggaran pemasaran untuk suatu produk baru adalah contoh dari keputusan semi terstruktur. Walaupun keputusan seperti ini biasanya tidak dapat secara penuh diotomatisasikan, namun sering didukung dari komputer (*computer-based decision*). Contoh keputusan jenis adalah investasi keuangan, pengevaluasian kredit, penjadwalan produksi, pemberian dana rehabilitasi sekolah, dan pengendalian persediaan.

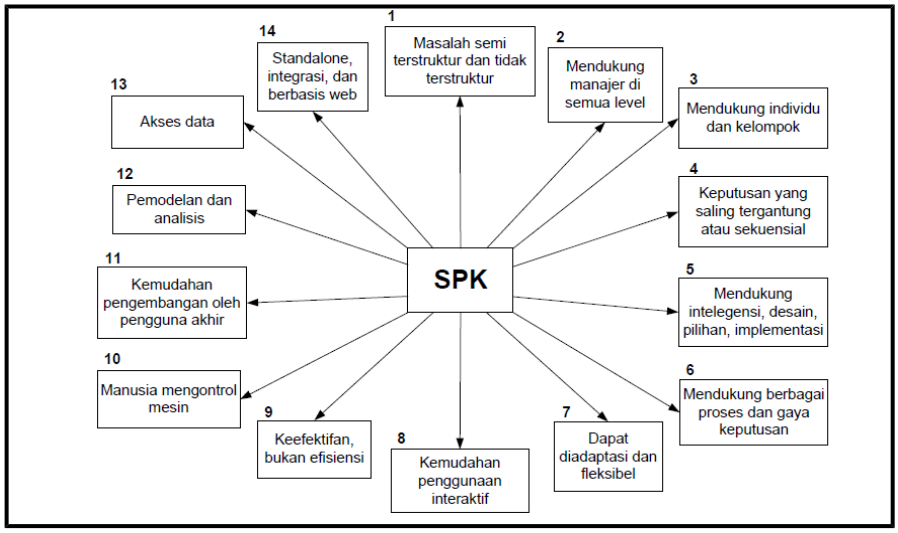
2.2.2 Sistem Pendukung Keputusan

Menurut McLeod dalam Pratiwi (2016:4), pengertian sistem pendukung keputusan menyatakan bahwa sistem pendukung keputusan merupakan sistem penghasil informasi yang ditujukan pada suatu masalah yang harus dibuat oleh manajer, sistem pendukung keputusan merupakan suatu sistem informasi yang ditujukan unyuk membantu manajemen dalam memecahkan masalah yang dihadapinya.

Definisi selengkapnya adalah sistem penghasil informasi spesifik yang ditujukan untuk memecahkan suatu masalah tertentu yang harus dipecahkan oleh manajer pada berbagai tingkatan. Sistem pendukung keputusan juga merupakan suatu sistem informasi berbasis komputer yang menghasilkan berbagai alternatif keputusan untuk membantu manajemen dalam menangani berbagai permasalahan yang terstruktur ataupun tidak terstruktur dengan menggunakan data atau model.

A. Karakteristik dan Kapabilitas DSS

Karakteristik dan kapabilitas kunci dari DSS adalah (ditunjukkan pada gambar 2.1) (Turban, E., dkk, 2005:140-143):



Gambar 2.1 Karakteristik dan kapabilitas kunci dari DSS

Sumber: *Turban, E., dkk (2005:142)*

1. Dukungan untuk pengambil keputusan, terutama pada situasi semistruktur dan tak terstruktur, dengan menyertakan penilaian manusia dan informasi terkomputerisasi. Masalah-masalah tersebut tidak dapat dipecahkan (atau tidak dapat dipecahkan dengan konvenien) oleh sistem komputer lain atay oleh metode atau alat kuantitatif standar.
2. Dukungan untuk semua level manajerial, dari eksekutif puncak sampai manajer lini.
3. Dukungan untuk individu dan kelompok. Misalnya yang kurang terstruktur sering memerlukan keterlibatan individu dari departemen dan tingkat organisasional yang berbeda atau bahkan dari organisasi lain. DSS mendukung tim virtual melalui alat-alat Web kolaboratif.
4. Dukungan untuk keputusan independen dan atau sekuensial. Keputusan dapat dibuat satu kali, beberapa kali, atau berulang (dalam interval yang sama).
5. Dukungan di semua fase proses proses pengambilan keputusan: inteligensi, desain, pilihan dan implementasi.
6. Dukungan di berbagai proses dan gaya pengambilan keputusan.
7. Adaptivitas sepanjang waktu. Pengambil keputusan seharusnya reaktif, dapat menghadapi perubahan kondisi secara cepat, dan dapat mengadaptasikan DSS untuk memenuhi perubahan tersebut. DSS bersifat fleksibel dan karena itu

pengguna dapat menambahan, menghapus, menggabungkan, mengubah, atau menyusun kembali elemen-elemen dasr. DSS juga fleksibel dalam hal dapat dimodifikasi untuk memecahkan masalah lain yang sejenis.

8. Pengguna merasa seperti rumah. Ramah-pengguna, kapabilitas grafis yang sangat kuat, dan antarmuka manusia mesin interaktif dengan satu bahasa alami dapat sangat meningkatan keefektifan DSS. Kebanyakan aplikasi DSS yang baru menggunakan antarmuka berbasis-Web.
9. Peningkatan terhadap keefektifan pengambilan keputusan (akurasi, timeliness, kualitas) ketimbang pada efisiensinya (biaya pengambilan keputusan). Ketika DSS disebarkan, pengambilan keputusan sering membutuhkan waktu lebih lama, namun keputusan lebih baik.
10. Kontrol penuh oleh pengambil keputusan terhadap semua langah proses pengambilan keputusan dalam langkah memecahkan suatu masalah. DSS secara khusus menekankan utnuk mendukung pengambilan keputusa, bukannya menggantikan.
11. Pengguna kahir dapat mengembangkan dan memodifikasi sendiri sistem sederhana. Sistem yang lebih besar dapat dibangun dengan bantuan ahli sistem informasi. Perangkat lunak OLAP dalam kaitannya dengan data warehouse membolehkan pengguna untuk membangun DSS yang cukup besar dan kompleks.
12. Biasanya model-model digunakan untuk menganalisis situasi pengambilan keputusan. Kapabilitas pemodelan memungkinkan eksperimen dengan berbagai strategi yang berbeda dibawah konfigurasi yang berbeda.
13. Akses disediakan untuk berbagai sumber data, format, dan tipe, mulai dari sistem informasi geografis (GIS) sampai sistem berorientasi-objek.
14. Dapat dilakukan sebagai alat standalone yang digunakan oleh seorang pengambil keputusan pada satu lokasi dan di beberapa organisasi sepanjang rantai persediaan. Dapat diintergrasikan dengan DSS lain dan atau aplikasi lain, dan dapat didistribusikan secara internal dan eksternal dengan menggunakan networking dan teknologi Web.

B. Komponen-komponen Sistem Pendukung Keputusan

Menurut Latif (2018:4), komponen-komponen Sistem Pendukung Keputusan terdiri dari:

1. *Data Management*

   Data Management. Termasuk database, yang mengandung data yang relevan untuk berbagai situasi dan diatur oleh software yang disebut *Database Management Systems* (DBMS).

2. *Model Management*

   Model Management. Melibatkan model finansial, statistikal, *management science*, atau berbagai model kuantitatif lainnya, sehingga dapat memberikan ke sistem suatu kemampuan analitis, dan manajemen *software* yang diperlukan.

3. *Communication (dialog subsystem)*

   Communication (dialog subsystem). User dapat berkomunikasi dan memberikan perintah pada DSS melalui sub sistem ini. Ini berarti menyediakan antarmuka.

4. *Knowledge Management*

   Knowledge Management. Subsistem optional ini dapat mendukung sub sistem lain atau bertindak sebagai komponen yang berdiri sendiri.

C. Fase Pengambilan Keputusan

Menurut Simon, ada tiga fase dalam proses Pengambilan Keputusan diantaranya sebagai berikut (Latif, 2018:5-6):

1. *Intellegence*

   Tahap ini merupakan proses penelusuran dan pendeteksian dari ruang lingkup problematika secara proses pengenalan masalah. Data masukan diperoleh, diproses dan diuji dalam rangka mengidentifikasi masalah.

2. *Design*

   Tahap ini merupakan proses menemukan mengembangkan dan menganalisis alternatif tindakan yang bisa dilakukan. Tahap ini meliputi menguji kelayakan solusi.

3. *Choice*

Pada tahap ini dilakukan proses pemilihan diantara berbagai alternatif tindakan yang mungkin dijalankan. Hasil pemilihan tersebut kemudian diimplementasikan dalam proses pengambilan keputusan.

2.3 Metode *Simple Multi Attribute Rating Technique* (SMART)

Menurut Goodwin dan Wright dalam Nofriansyah dan Defit (2017:27), "Metode Smart merupakan metode perbandingan kuantitatif yang digunakan untuk mengkombinasikan ketidaksamaan perbandingan.

Model yang digunakan dalam *Simple Multi Attribute Rating Technique* (SMART) yaitu:

$$(a_i) = \sum w_{j=1}^{m} \; Wj \; Ui \; (ai)$$

Keterangan:

1. W_j = Nilai Pembobotan Kriteria ke- j dan K- kriteria
2. $U(a_i)$ = nilai Utility kriteria ke-I untuk kriteria ke-i

Dimana i= 1,2,…….m

Adapun algoritma penyelesaian dari Metode *Simple Multi Attribut Rating Technique* (SMART) yaitu sebagai berikut:

1. Langkah 1 : Menentukan Jumlah Kriteria dari Keputusan yang akan di ambil
2. Langkah 2 : Sistem secara default memberikan nilai 0-100 berdasarkan prioritas dengan melakukan normalisasi ($W_j / \sum W_j$)
3. Langkah 3 : Memberikan nilai kriteria untuk setiap alternatif
4. Langkah 4 : Menghitung nilai Utility untuk setiap kriteria masing-masing

$$u_i(a_i) = 100 \frac{C_{max} - C_{out\ i}}{C_{max} - C_{min}} \%$$.....

Dimana:

1) $u_i(a_i)$adalah nilai utiliti kriteria ke-1 untuk kriteria ke–I,

2) C_{max} adalah nilaikriteriamaksimal

3) C_{min} adalahnilai kriteriaminimal

4) $C_{out}\ ^{i}$ adalah nilai kriteria ke-i.

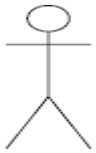
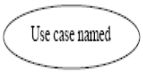
5. Langkah 5 : Menghitung nilai akhir dan melakukan Perangkingan

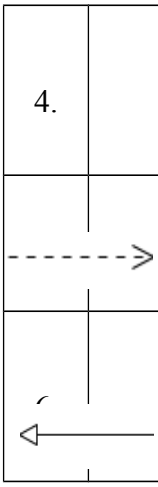
2.4 Teori yang Berhubungan Teknik Analisa yang Digunakan

2.4.1 *Use Case*

Use Case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Sukamto dan Shalahuddin, 2017). Simbol-simbol *Use Case* Diagram dapat dilihat pada Tabel 2.1

Tabel 2.1 Simbol Use Case Diagram

No	Simbol	Nama	Keterangan
1.		Aktor/ *Actor*	Aktor adalah pengguna sistem. aktor tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan input atau memberikan output, maka aplikasi tersebut juga bisa dianggap sebagai aktor.
2.		*Use case*	*Use case* digambarkan sebagai lingkaran elips dengan nama *use case* dituliskan didalam elips tersebut, digunakan sebagai unit-unit yang saling bertukat pesan antar unit / aktor.
3.		*Association* / Asosiasi	Asosiasi digunakan untuk menghubungkan *actor* dengan *use case*. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara *Actor* dengan *Use Case*.
		Generalizati	Hubungan dimana objek anak

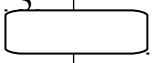
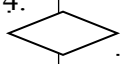
4.		*on* / Generalisasi	(*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
		Include	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
		Extend / Ekstensi	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan

(Sumber: Sukamto dan Shalahuddin, 2017)

2.4.2 *Activity Diagram*

Menurut Sukamto dan Shalahuddin dalam Aprianti (2016:23), *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas menggambarkan aktivitas –aktivitas sistem bukan apa yang dilakukan aktor.

Tabel 2.2 Simbol *Activity Diagram*

No	Simbol	Nama	Keterangan
1.		*Start state*	Titik awal atau permulaan
2.		*End state*	Titik akhir atau akhir dari aktivitas
3.		*Activity*	*Activity* atau aktivitas yang dilakukan oleh*actor*
4.		*Decision*	Pilihan untuk mengambil keputusan
5.		*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.

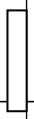
(Sumber: Sukamto dan Shalahuddin, 2017)

2.4.3 *Sequence Diagram*

Menurut Pratama dalam Sukamto dan Shalahuddin, 2017, "*Sequence* diagram menggambarkan *sequence* (aliran) pengiriman pesan *(message)* yang terjadi di aplikasi, sebagai bentuk interaksi dengan pengguna *(user)*".

Adapun simbol-simbol yang digunakan dalam *sequence* diagram adalah sebagai berikut:

Tabel 2.3 Simbol *Sequence* Diagram

No.	Gambar	Nama	Keterangan
1.		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2.		Lifeline	Menyatakan kehidupan suatu objek.
3.		Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
4.		Pesan tipe call	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.

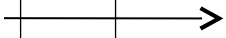
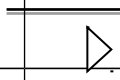
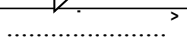
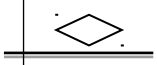
(Sumber: Sukamto dan Shalahuddin, 2017)

2.4.4 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendifinisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Simbol-simbol yang ada pada *Class diagram* ditunjukan oleh Tabel 2.4.

Tabel 2.4 *Simbol Class diagram*

No.	Gambar	Nama	Keterangan

1.	Nama_kelas +atribut +operasi()	*Class/Kelas*	Kelas pada struktur system
2.		*Interface*/Antarmuka	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3.		*Association*/Asosiasi	Relasiantarkelas dengan makna umum,asosiasi biasanya disertai dengan *multiplicity.*
4.		*Directed association*/Asosiasi berarah	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5.		Generalisasi	Relasi antar kelas dengan makna generalisasi- spesialisasi (umum khusus).
6.		*Dependency*/ Kebergantungan	Relasi antar kelas dengan makna kebergantungan antar kelas
7.		*Aggregation*/ Agregasi	Relasi antar kelas dengan makna semua-bagian.

(Sumber: Sukamto dan Shalahuddin, 2017)

2.5 Teori Pendukung Lainnya

2.5.1 Database

Basis data (*database*) dapat didefinisikan sebagai himpunan kelompok data saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah (Hidayatullah dan Kawistara, 2014:147).

2.5.2 HTML (*Hypertext Markup Language*)

Hypertxt Markup Language (HTML) adalah bahasa standard yang digunakan untuk menampilkan halaman *web*. Yang bisa dilakukan HTML yaitu (Hidayatullah dan Kawistara, 2014:13).

1. Mengatur tampilan dari halaman *web* dan isinya.
2. Membuat tabel dalam halamn *web*.
3. Mempublikasikan halaman *web* secara *online*.
4. Membuat form yang bisa digunakan untuk menangani registrasi dan transaksi via *web*.
5. Menambahkan objek-objek seperti citra, audio, video, animasi, *java applet* dalam halaman *web*.
6. Menampilkan area gambar (canvas) di *browser*.

2.5.3 MySQL

MySQL adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrogram aplikasi web. Kelebihan dari *MySQL* adalah gratis, handal, selalu di-*update* dan banyak form yang memfasilitasi para pengguna jika memiliki kendala. *MySQL* juga menjadi DBMS yang sering dibundling dengan web *server* sehingga proses instalasinya jadi lebih mudah (Hidayatullah dan Kawistara, 2014: 180).

2.5.4 PHP (*Hypertext Preprocessor)*

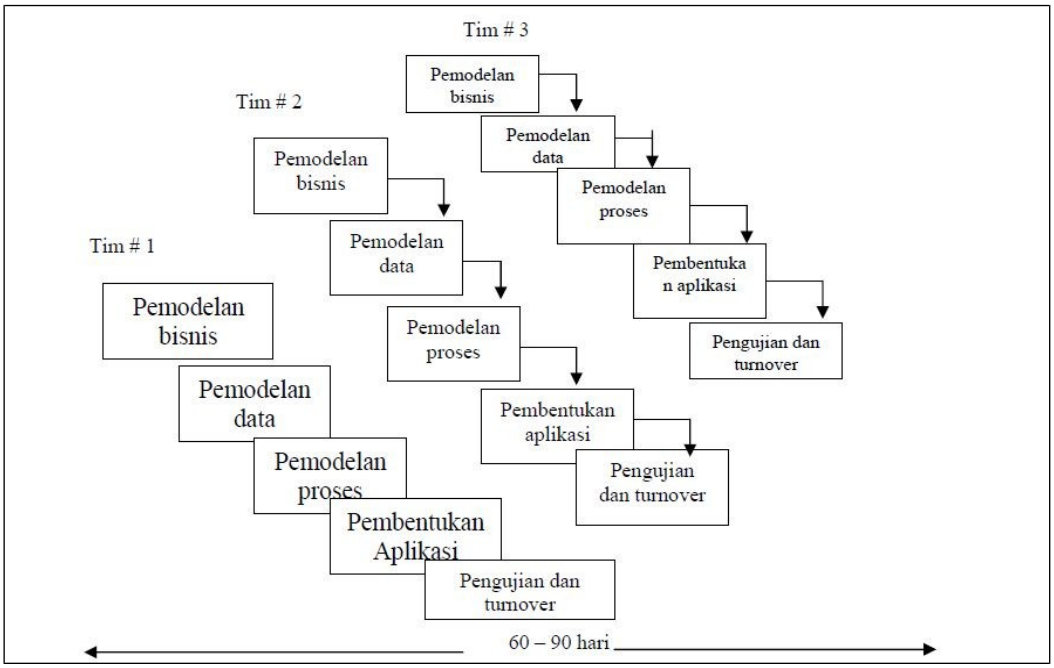
PHP *Hypertext Preprocessor* atau disingkat dengan PHP ini adalah suatu bahasa *scripting* khusunya digunakan untuk web *development*. Karena sifatnya yang *server side scripting*, maka untuk menjalankan PHP harus menggunakan *web server* (Hidayatullah dan Kawistara, 2014: 231).

2.6 Metode Pengembangan *Rapid Application Development* (*RAD*)

Sistem pendukung keputusan untuk penetuan kendaraan hilang pada Polres Merangin Menggunakan Model *Rapid Application Development* (*RAD*), merupakan salah satu metode pengembangan software yang termasuk dalam *Software Development Life Cycle*.

Menurut Sukamto dan Shalahuddin (2018:34), *Rapid Application Development* (RAD) adalah model proses pengembangan perangkat lunak yang bersifat inkremental terutama untuk waktu pengerjaan yang pendek. Model RAD adalah adaptasi dari model air terjun versi kecepatan tinggi dengan menggunakan model air terjun untuk pengembangan setiap komponen perangkat lunak.

Jika kebutuhan perangkat lunak dipahami dengan baik dan lingkup perangkat lunak dibatasi dengan baik sehingga tim dapat menyelesaikan pembuatan perangkat lunak dengan waktu yang pendek. Model RAD membagi tim pengembang menjadi beberapa tim untuk mengerjakan beberapa komponen masing-masing tim pengerjaan dapat dilakukan secara paralel. Berikut adalah gambar dari model RAD:



Gambar 2.2 Model *RAD*

Sumber: *Sukamto dan Shalahuddin (2018:34)*

1. Pemodelan Bisnis

   Pemodelan yang dilakukan untuk memodelkan funsgi bisnis untuk mengetahui informasi apa yang terkait proses bisnis, informasi apa saja yang harus dibuat, siapa yang harus membuat informasi itu, bagaimana alur informasi itu, proses apa saja yang terkait informasi itu.

2. Pemodelan Data
   Memodelkan data apa saja yang dibutuhkan berdasarkan pemodelan bisnis dan mendefinisikan atribut-atributnya beserta relasinya dengan data-data yang lain.
3. Pemodelan Proses
   Mengimplementasikan fungsi bisnis yang sudah didefinisikan terkait dengan pendefinisian data.
4. Pembentukan Aplikasi
   Mengimplementasikan pemodelan proses dan data menjadi program. Model *RAD* sangat menganjurkan pemakaian komponen yang sudah ada jika dimungkinkan.

5. Pengujian dan Turnover
   Menguji komponen-komponen yang dibuat. Jika sudah teruji maka tim pengembang komponen dapat beranjak untuk mengembangkan komponen berikutnya.

Model RAD sangat cocok diterapkan apabila memenuhi kriteria proyek sebagai berikut menurut Sukamto dan Shalahuddin (2018:37):

1. Anggota tim sudah berpengalaman mengembangkan perangkat lunak yang sejenis.
2. Pengembangan sudah memiliki komponen-komponen sistem yang bisa digunakan kembali dalam proyek tersebut.

2.7 Metode Pengujian *Black Box*

Metode pengujian digunakan untuk mengetahui fungsi yang telah ditentukan bahwa suatu sistem telah dirancang dapat menunjukan bahwa masing-masing fungsi sepenuhnya beroperasi. Pengujian kotak hitam *(black box),* juga disebut pengujian prilaku, berfokus pada persyaratan fungsional perangkat lunak.

Artinya, teknik pengujian kotak hitam memungkinkan untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program. Pengujian kotak hitam *(black box)* bukan teknik alternatif untuk kotak putih *(white box).*

Pengujian kotak hitam *(black box)* berupaya untuk menemukan kesalahan dalam kategori berikut: (1) fungsi yang salah atau hilang, (2) kesalahan dalam

struktur data atau akses basis eksternal, (4) kesalahan perilaku atau kinerja, dan (5) kesalahan inisialisasi dan penghentian (Roger S. Pressman, 2012:597).