BAB II
LANDASAN TEORI

2.1 Sistem Pendukung Keputusan

Sistem pendukung keputusan (SPK) adalah sistem interaktif yang membantu proses pengambilan keputusan dengan menyajikan alternatif-alternatif yang berasal dari pengolahan data, informasi, dan rancangan model. SPK menggabungkan sumber kecerdasan individu dengan kemampuan komponen untuk meningkatkan kualitas keputusan. Sistem pendukung keputusan bukan alat untuk mengambil keputusan, melainkan sebuah sistem yang membantu pengambil keputusan dengan menyediakan informasi yang relevan dari data yang telah diolah, sehingga memungkinkan pengambilan keputusan tentang suatu masalah dapat dilakukan lebih cepat dan akurat (Pami, 2017). SPK bertujuan untuk menyediakan informasi, memberikan panduan, melakukan prediksi, serta mengarahkan pengguna informasi agar dapat mengambil keputusan dengan lebih baik (Darpi Nurhayati, 2022).

Berikut adalah beberapa elemen kunci yang mendefinisikan sistem pendukung keputusan :

A. Data : SPK mengintegrasikan dan menyimpan data dari berbagai sumber untuk menyediakan dasar informasi bagi pengambil keputusan. Data ini bisa berupa data histori, data saat ini, atau data proyeksi.
B. Model : SPK sering kali memanfaatkan model matematika atau statistik untuk menganalisis data dan menghasilkan prediksi atau simulasi. Model-model ini membantu dalam memahami implikasi dari berbagai keputusan yang mungkin diambil.
C. Kriteria dan Alternatif : Dalam proses pengambilan keputusan, pengguna SPK dapat menentukan kriteria yang relevan dan alternatif yang mungkin. Kriteria adalah faktor atau indikator yang digunakan untuk menilai alternatif, sementara alternatif adalah pilihan atau Tindakan yang dapat diambil.

D. Antarmuka Pengguna : SPK menyediakan antarmuka yang digunakan oleh pengguna untuk memasukkan data, menjalankan analisis, dan mengeksplorasi hasil. Antarmuka ini harus mudah digunakan dan intuitif agar pengguna dapat dengan cepat memahami informasi yang diberikan.
E. Proses Pengambilan Keputusan : SPK membantu pengguna dalam melalui langkah-langkah proses pengambilan keputusan, seperti identifikasi masalah, pemodelan, analisis, evaluasi alternatif, dan pemilihan keputusan terbaik.
F. Dukungan Interaktif : Salah satu aspek utama dari SPK adalah interaktivitasnya. Pengguna dapat bereksperimen dengan berbagai skenario, mengubah kriteria, dan melihat dampaknya pada hasil keputusan secara langsung.
G. *Reporting* dan Visualisasi : SPK dapat menghasilkan laporan, grafik, dan visualisasi data yang membantu pengguna memahami informasi dengan lebih baik. Ini membantu dalam menjelaskan konsep-konsep yang kompleks atau menyoroti pola penting dalam data.

2.2 Pegawai Negeri Sipil

Pegawai Negeri Sipil (PNS) adalah sebutan untuk seseorang yang diangkat sebagai pekerja di instansi pemerintah dalam suatu negara. PNS biasanya dipekerjakan oleh pemerintah pusat atau daerah untuk melaksanakan tugas dan tanggung jawab tertentu sesuai dengan bidang masing-masing dalam struktur organisasi. Dalam menjalankan tugas mereka, pegawai mengekspresikan kreativitas mereka sesuai dengan panduan yang telah ditetapkan oleh pimpinan atau atasan mereka (Sihaloho et al., 2022).

2.3 Kriteria dan Bobot

Kriteria adalah standar atau karakteristik yang digunakan untuk menilai atau mengukur sesuatu. Dalam berbagai konteks, kriteria digunakan untuk membaut keputusan atau evaluasi terhadap suatu hal. Adapun Bobot merujuk pada nilai relatif atau pentingnya suatu elemen dalam suatu konteks tertentu. Dalam konteks penilaian atau pengambilan keputusan, bobot dapat digunakan

untuk memberikan tingkat kepentingan yang berbeda pada setiap kriteria atau elemen yang dinilai. Bobot menggambarkan sejauh mana suatu faktor memiliki dampak terhadap hasil akhir atau keputusan. Kriteria dan bobot diperlukan untuk menentukan siapa yang akan dinilai sebagai pegawai terbaik (Sihaloho et al., 2022).

2.4 Metode *Weighted Product*

Metode *Weighted Product* (WP) adalah salah satu metode yang digunakan untuk penyelesaian sistem pengambilan keputusan dengan mempertimbangkan kriteria serta bobotnya (Fridayanthie et al., 2020). Metode *weighted product* menggunakan perkalian untuk menghubungkan *ratting* atribut (nilai dari alternatif), di mana *ratting* setiap atribut harus dipangkatkan terlebih dahulu dengan bobot atribut yang bersangkutan. Proses ini mirip dengan proses normalisasi (Nurjannah et al., 2015).

Langkah-langkah umum dalam metode *weighted product* adalah sebagai berikut :

A. Penentuan Kriteria : Identifikasi kriteria yang akan digunakan untuk menilai alternatif. Kriteria ini dapat berupa faktor-faktor yang relevan dengan masalah atau keputusan yang sedang dihadapi.
B. Penentuan Bobot : Berikan bobot pada masing-masing kriteria untuk menunjukkan tingkat pentingnya dalam pengambilan keputusan. Bobot ini mencerminkan preferensi atau prioritas pengambil keputusan terhadap setiap kriteria.
C. Normalisasi Nilai : Nilai-nilai kriteria untuk setiap alternatif sering kali perlu dinormalisasi agar dapat dibandingkan dengan mudah. Normalisasi dilakukan untuk menghindari masalah perbandingan nilai yang berbeda skala.
D. Perhitungan *Weighted Product* : Kalikan nilai-nilai kriteria dengan bobot yang sesuai untuk setiap alternatif, kemudian jumlahkan hasilnya untuk setiap alternatif.

E. Pemilihan Alternatif Terbaik : Alternatif dengan hasil terbesar setalah perhitungan *weighted product* adalah pilihan terbaik menurut kriteria yang telah ditetapkan.

Metode *weighted product* dalam proses perhitungannya dapat disingkat, yang terdiri dari 3 langkah. Berikut adalah langkah-langkahnya (Hafiz, 2018) :

1) Perbaikan bobot kriteria, dengan persamaan berikut :

$$W_j = \frac{W_j}{\sum wj} \qquad \text{…….. (2.1)}$$

2) Menghitung vektor S. Langkah ini sama seperti proses normalisasi, dengan persamaan berikut :

$$S_i = \prod_{j=1}^{n} X_{ij}^{W_j} \qquad \text{…….. (2.2)}$$

Di mana $\sum wj = 1$. wj adalah pangkat bernilai positif untuk kategori kriteria keuntungan dan pangkat bernilai negatif untuk kategori biaya/*cost*.

3) Menghitung vektor V, atau preferensi relatif dari setiap alternatif, untuk perangkingan dengan persamaan berikut :

$$V_i = \frac{\prod_{j=1}^{n} X_{ij}^{W_j}}{\prod_{j=1}^{n} (X_j)^{W_j}} \qquad \text{…….. (2.3)}$$

Sederhananya seperti :

$$V_i = \frac{Si}{\sum Si}$$

Keterangan :

S = Preferensi alternatif, dianalogikan sebagai vektor S.

V = Preferensi alternatif, dianalogikan sebagai vektor V.

X = Nilai kriteria.

Wj = Bobot kriteria.

i = Alternatif.

j = Kriteria.

n = Banyaknya kriteria.

$*$ = Banyaknya kriteria yang telah dinilai pada vektor S.

$\sum Wj$ = Penjumlahan bobot kriteria.

2.5 Basis Data

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis dapat diartikan sebagai markas, gudang, tempat berkumpul. Sedangkan data adalah fakta yang mewakili suatu objek seperti manusia, barang, hewan, peristiwa, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Basis data sendiri dapat didefinisikan dalam jumlah sudut pandang seperti (Yanto, 2016, p. 10) :

A. Himpunan kelompok data yang saling berhubungan yang diorganisasi sedemikian rupa agar dapat dimanfaatkan Kembali dengan cepat dan mudah.
B. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi), untuk memenuhi berbagai kebutuhan.
C. Kumpulan *file* yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

2.5.1 *Database Management System* (DBMS)

Sebuah DBMS adalah data yang saling berhubungan yang dikelompokkan dalam sebuah tabel atau beberapa tebal dan sebuah aplikasi program yang mengatur cara mengakses data tersebut. Kumpulan dari data tersebut biasanya disebut basis data, yang berisikan informasi yang nyata untuk sebuah Perusahaan. Tujuan utama DBMS adalah menyediakan sebuah cara untuk menyimpan dan mengambil informasi basis data secara efisien dan nyaman. Manajemen data meliputi struktur informasi penyimpanan dan mekanisme untuk memanipulasi informasi yang ada dalam basis data (Widodo, 2017, p. 3).

2.5.2 Data

Data tersusun dari fakta-fakta mentah yakni fakta-fakta belum diproses untuk mengungkapkan maknanya. Sebagai contoh data yang diinput pada sebuah *form* aplikasi penjualan. Di sana mungkin terdapat inputan untuk kolom *id* produk, nama produk, harga stok, kategori, dan sebagainya. Semua yang Anda *input* di sini adalah data. Bila data yang diinput ke dalam *form* lalu disimpan, maka ia ditempatkan ke dalam basis data (Hadiprakoso, 2021, pp. 4–5). Berikut adalah contoh data yang dikelompokkan dalam sebuah tabel :

Tabel 2.1 Contoh Data Dalam Tabel

***Id* Produk**	**Nama Produk**	**Harga Produk**	**Kategori**
Pj000001	Gelas Kaca	Rp. 12,000	Gelas
Pj000002	Piring Kaca	Rp. 20,000	Piring
Pj000003	Piring Plastik	Rp. 11,000	Piring

2.5.3 MySQL

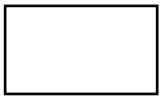
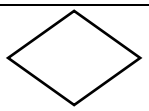
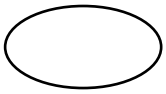
MySQL adalah DBMS yang *open source* dengan dua bentuk lisensi GNU *General Public License* (GPL), yaitu *Free Software* (perangkat lunak bebas) dan *Shareware* (perangkat lunak berpemilik yang penggunanya terbatas). Seperti yang sudah disebutkan sebelumnya, *MySQL* masuk ke dalam jenis RDBMS (*Relational Database Management System*). Maka dari itu istilah semacam baris, kolom, tabel, dipakai pada *MySQL*. Contohnya di dalam *MySQL* sebuah *database* terdapat satu atau beberapa tabel. *MySQL* juga mendukung bahasa *database SQL* sebagai bahasa interaktif dalam mengelola data (Fitri, 2020, p. 2).

2.5.4 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram adalah suatu diagram untuk menggambarkan desain konseptual dari model konseptual suatu basis data relasional. ERD juga merupakan gambaran yang merelasikan antara objek

yang satu dengan objek yang lain dari objek di dunia nyata yang sering dikenal dengan hubungan antar entitas (Yanto, 2016, p. 32).

Tabel 2.2 Simbol ERD

Simbol	Keterangan
	Entitas, yaitu kumpulan dari objek yang dapat diidentifikasikan secara unik.
	Relasi, yaitu hubungan yang terjadi antara satu atau lebih entitas.
	Atribut, yaitu karakteristik dari *entity* atau relasi yang merupakan penjelasan detail tentang entitas.
	Hubungan antara *entity* dengan atributnya dan himpunan entitas dengan himpunan relasinya.

2.5.5 Normalisasi Data

Normalisasi adalah pendekatan alternatif dalam pengembangan desain logika basis data relasional yang tidak terkait langsung dengan model data. Metode ini melibatkan penerapan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal (Monica et al., 2015).

Normalisasi data dimulai dengan sebuah tabel yang mengandung banyak data dan kompleksitas. Tabel ini kemudian diorganisir menjadi beberapa tabel yang lebih kecil dengan struktur data yang lebih stabil. Terdapat tiga langkah dalam normalisasi data, yaitu (Iskandar et al., 2011):

A. Menghilangkan kelompok berulang/*repeating groups*.
B. Menghilangkan ketergantungan parsial/*partial dependencies*.
C. Menghilangkan ketergantungan trasitif/*transitive dependencies*.

Berikut adalah contoh studi kasus dalam normalisasi data, dan langkah-langkahnya sebagai berikut :

1) Bentuk Tidak Normal

Bentuk tidak normal adalah suatu bentuk di mana semua data dikumpulkan apa adanya tanpa mengikuti aturan-aturan tertentu. Bisa jadi data yang dikumpulkan akan tidak lengkap dan terjadi duplikasi data. Berikut contohnya :

Tabel 2.3 Bentuk Tidak Normal (Unnormalize)

NoProyek	NamaProyek	NoPegawai	NamaPegawai	Golongan	BesarGaji
NP001	BRR	Peg01	Anton	A	1.000.000
		Peg02	Paula	B	900.000
		Peg06	Koko	C	750.000
NP002	PEMDA	Peg01	Anton	A	1.000.000
		Peg12	Sita	B	900.000
		Peg14	Yusni	B	900.000

Untuk mendapatkan hasil yang paling normal, proses normalisasi dimulai dengan normalisasi pertama. *Field-field* di atas yang merupakan group berulang adalah NoPegawai, NamaPegawai, Golongan, dan BesarGaji.

2) Bentuk Normal Pertama

Solusinya adalah menghilangkan duplikasi dengan mencari ketergantungan parsial, menjadikan beberapa *field* bergantung pada satu atau beberapa *field* lainnya. Karena kunci yang dapat dijadikan kunci adalah NoProyek dan NoPegawai, langkah selanjutnya adalah menentukan *field-field* mana yang bergantung pada NoProyek dan mana yang bergantung pada NoPegawai. Contoh normalisasi bentuk pertama dapat dilihat pada table berikut :

Tabel 2.4 Normalisasi Bentuk 1NF

NoProyek	NamaProyek	NoPegawai	NamaPegawai	Golongan	BesarGaji
NP001	BRR	Peg01	Anton	A	1.000.000
NP001	BRR	Peg02	Paula	B	900.000

NP001	BRR	Peg06	Koko	C	750.000
NP002	PEMDA	Peg01	Anton	A	1.000.000
NP002	PEMDA	Peg12	Sita	B	900.000
NP002	PEMDA	Peg14	Yusni	B	900.000

3) Bentuk Normal Kedua

Field-field yang bergantung pada satu *field* harus dipisahkan dengan jelas, misalnya NoProyek menjelaskan NamaProyek, dan NoPegawai menjelaskan NamaPegawai, Golongan, serta BesarGaji. Contoh normalisasi bentuk kedua dapat dilihat pada tabel berikut :

Tabel 2.5 Normalisasi Bentuk Kedua Tabel Proyek

NoProyek	**NamaProyek**
NP001	BRR
NP002	PEMDA

Tabel 2.6 Normalisasi Bentuk Kedua Tabel Pegawai

NoPegawai	**NamaPegawai**	**Golongan**	**BesarGaji**
Peg01	Anton	A	1.000.000
Peg02	Paula	B	900.000
Peg06	Koko	C	750.000
Peg12	Sita	B	900.000
Peg14	Yusni	B	900.000

Untuk membuat hubungan antara dua tabel, dibuat suatu tabel yang berisi *key-key* dari tabel yang lain.

Tabel 2.7 Normalisasi Bentuk Kedua Tabel Proyek Pegawai

NoProyek	**NoPegawai**
NP001	Peg01
NP001	Peg02
NP001	Peg06

NP002	Peg01
NP002	Peg12
NP002	Peg14

4) Bentuk Normal Ketiga

Pada tabel di atas, masih terdapat masalah di mana BesarGaji tergantung pada Golongan. Padahal, Golongan bukan merupakan *field* kunci. Oleh karena itu, kita perlu memisahkan *field* non-kunci, yaitu Golongan dan BesarGaji yang sebelumnya memiliki ketergantungan transitif pada field kunci NoPegawai, untuk menghilangkan ketergantungan tersebut. Contoh normalisasi bentuk ketiga dapat dilihat pada tabel berikut :

Tabel 2.8 Normalisasi Bentuk Ketiga Tabel Proyek

NoProyek	**NamaProyek**
NP001	BRR
NP002	PEMDA

Tabel 2.9 Normalisasi Bentuk Ketiga Tabel Proyek Pegawai

NoProyek	**NoPegawai**
NP001	Peg01
NP001	Peg02
NP001	Peg06
NP002	Peg01
NP002	Peg12
NP002	Peg14

Tabel 2.10 Normalisasi Bentuk Ketiga Tabel Pegawai

NoPegawai	**NamaPegawai**	**Golongan**
Peg01	Anton	A
Peg02	Paula	B

Peg06	Koko	C
Peg12	Sita	B
Peg14	Yusni	B

Tabel 2.11 Normalisasi Bentuk Ketiga Tabel Golongan

Golongan	**BesarGaji**
A	1.000.000
B	900.000
C	750.000

2.6 PHP

PHP (*Hypertext Preprocessor*) yaitu bahasa pemrograman web *server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada server (*server-side* HTML *embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru/*up to date*. Semua *script* PHP dieksekusi pada server di mana *script* tersebut dijalankan (Anhar, 2010, p. 3).

2.7 Framework Codeigniter

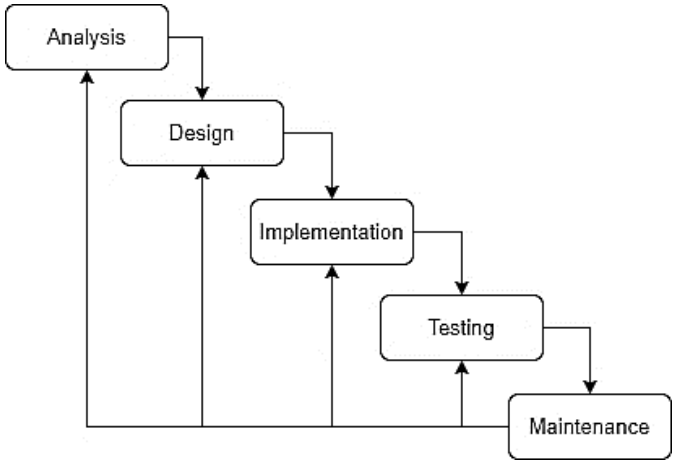
Codeigniter atau sering disingkat CI merupakan salah satu *framework* yang digunakan untuk membuat aplikasi web berbasis PHP. *Framework* CI bersifat *open source* dan memiliki fungsionalitas yang sangat kaya sehingga dapat membantu mempercepat para pengembang aplikasi web dalam bekerja. *Framework* CI hadir dengan konsep MVC (*Model View Controller*) yang mempermudah pengembang dalam mengelola proyek. *Model* merupakan bagian yang terkait dengan interaksi program dan *database*. Adapun *View* merupakan bagian yang berkaitan dengan antarmuka atau tampilan dari aplikasi web. Sedangkan *Controller* merupakan bagian yang berkaitan dengan fungsi-fungsi yang menghubungkan *database* dengan antarmuka program,

atau fungsi-fungsi logika yang diterapkan pada aplikasi web (Wadi, 2020, p. 2).

2.8 Model Pengembangan Sistem

2.8.1 *Waterfall*

Model air terjun (*Waterfall* Model) merupakan pendekatan klasik dalam pengembangan perangkat lunak yang menggambarkan metode pengembangan linier dan berurutan. Ini terdiri dari lima hingga tujuh fase, setiap fase didefinisikan oleh tugas dan tujuan yang berbeda, di mana keseluruhan fase menggambarkan siklus hidup perangkat lunak hingga pengirimannya. Setelah fase selesai, langkah pengembangan selanjutnya mengikuti dan hasil dari fase sebelumnya mengalir ke fase berikutnya (Untari, 2020, pp. 21–22).



Gambar 2.1 Tahapan Model *Waterfall*

A. *Requirement* (analisis kebutuhan)

Mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Fase ini harus dikerjakan secara lengkap untuk bisa menghasilkan desain yang lengkap.

B. Design System

Dalam tahap ini pengembang akan menghasilkan sebuah sistem secara keseluruhan dan menentukan alur perangkat lunak hingga algoritma yang detail.

C. *Coding & Testing* (penulisan sinkode program/ *implementation*)

Tahapan di mana seluruh desain diubah menjadi kode-kode program. Kode program yang dihasilkan masih berupa modul-modul yang akan diintegrasikan menjadi sistem yang lengkap.

D. Integration & Testing

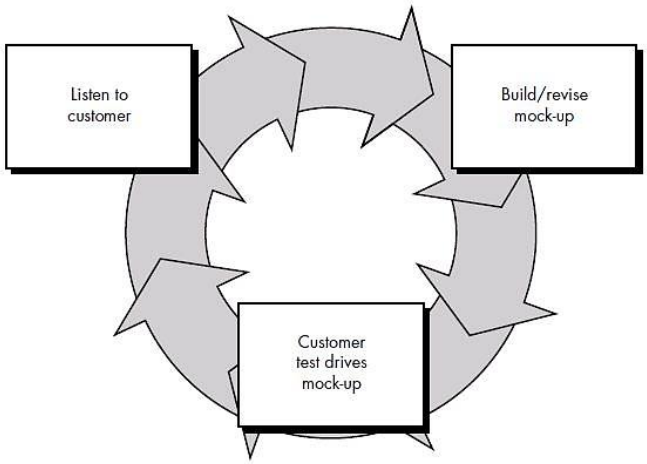
Di tahap ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian, ini dilakukan untuk mengetahui apakah *software* yang dibuat telah sesuai dengan desainnya dan fungsi pada *software* terdapat kesalahan atau tidak.

E. Operation & Maintenance

Yaitu instalasi dan proses perbaikan sistem sesuai dengan yang disetujui.

2.8.2 *Prototyping*

Prototyping adalah proses merancang sebuah prototipe, di mana prototipe itu sendiri adalah suatu model dari produk yang mungkin belum memuat semua fitur produk sesungguhnya, tetapi telah mencakup fitur-fitur utama dari produk tersebut. Prototipe umumnya digunakan untuk keperluan pengujian sebelum memasuki tahap pembuatan produk sesungguhnya. Dengan menggunakan metode *prototyping* ini, pengembang dan pelanggan dapat berinteraksi secara langsung selama proses pembuatan produk (Untari, 2020, pp. 23–24).



Gambar 2.2 Tahapan Model *Prototyping*

A. Mendengarkan Pelanggan

Pada tahap ini dilakukan pengumpulan kebutuhan dari sistem dengan cara mendengar keluhan dari pelanggan. Untuk membuat suatu sistem yang sesuai kebutuhan, maka harus diketahui terlebih dahulu bagaimana sistem yang sedang berjalan untuk kemudian mengetahui masalah yang terjadi.

B. Merancang dan Membuat *Prototype*

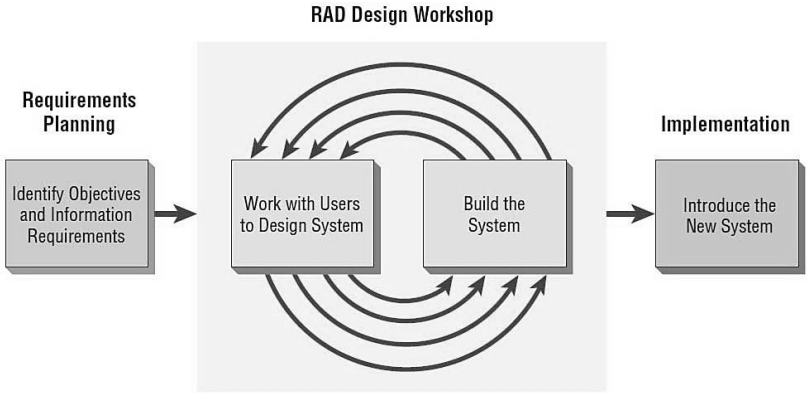
Pada tahap ini dilakukan perancangan dan pembuatan prototipe sistem. Prototipe yang dibuat disesuaikan dengan kebutuhan sistem yang telah didefinisikan sebelumnya dari keluhan pelanggan atau pengguna.

C. Uji Coba

Prototipe dari sistem diuji coba oleh pelanggan atau pengguna. Kemudian dilakukan evaluasi kekurangan-kekurangan dari kebutuhan pelanggan. Pengembangan kemudian Kembali mendengarkan keluhan dari pelanggan untuk memperbaiki prototipe yang ada.

2.8.3 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah proses pengembangan perangkat lunak tambahan model yang menekankan siklus pengembangan yang sangat singkat. Model RAD adalah adaptasi "kecepatan tinggi" dari model sekuensial linier yang berkembang pesat dicapai dengan menggunakan konstruksi berbasis komponen (Roger S. Pressman, 2001, p. 32).



Gambar 2.3 Tahapan Model RAD

A. *Requirement Planning*

Dalam tahap ini, diketahui apa saja yang menjadi kebutuhan sistem yaitu dengan mengidentifikasikan kebutuhan informasi dan masalah yang dihadapi untuk menentukan tujuan, batasan-batasan sistem, kendala, dan juga alternatif pemecahan masalah. Analisis digunakan untuk mengetahui perilaku sistem dan juga untuk mengetahui aktivitas apa saja yang ada dalam sistem tersebut.

B. *Design Workshop*

Tahap ini yaitu mengidentifikasi solusi alternatif dan memilih solusi yang terbaik. Kemudian membuat desain proses bisnis dan desain pemrograman untuk data-data yang telah didapatkan dan dimodelkan dalam arsitektur sistem informasi. *Tools* yang digunakan dalam pemodelan sistem biasanya menggunakan *unified modeling language* (UML).

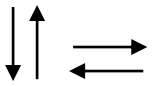
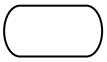
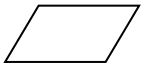
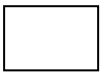
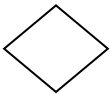
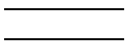
C. *Implementation*

Setelah *design workshop* dilakukan, selanjutnya sistem diimplementasikan (*coding*) ke dalam bentuk yang dimengerti oleh mesin yang diwujudkan dalam bentuk program atau unit program. Tahap implementasi sistem merupakan tahap meletakkan sistem supaya siap untuk dioperasikan.

2.9 *Flowmap*

Campuran antara peta dan *flowchart* biasanya disebut dengan istilah *flowmap. Flowmap* adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur dalam suatu program. Pembuatan *flowmap* biasanya dilakukan setelah tahap analisis. *Flowmap* menggambarkan alur dari sistem yang akan dibuat secara menyeluruh dan menjelaskan dengan rinci urutan langkahnya, mulai dari awal hingga akhir. Manfaat dari *flowmap* adalah membantu programmer dalam menganalisis alternatif lain dalam operasional dan memecahkan masalah ke dalam segmen yang lebih kecil (Sirait & Seabtian, 2019). Adapun simbol-simbol yang digunakan adalah sebagai berikut :

Tabel 2.12 Simbol *Flowmap*

Simbol	Nama	Keterangan
	Arah aliran	Menunjukkan arah aliran dokumen antar bagian yang terkait pada suatu sistem, baik dari sistem atau keluar sistem.
	Terminator	Menunjukkan permulaan (*start*) atau akhir (*stop*) dari suatu kegiatan.
	*Input*/*Output*	Mewakili proses *input*/*output* data tanpa tergantung jenis peralatannya.
	Proses Manual	Menunjukkan proses yang dilakukan secara manual.
	Proses Komputer	Menunjukkan proses yang dilakukan secara komputerisasi.
	Penghubung	Menunjukkan aliran dokumen yang terputus atau terpisah pada *flowmap* yang sama.
	Penghubung antar *flowmap*	Menunjukkan aliran dokumen yang saling berhubungan pada *flowmap* yang berbeda.
	Pengarsipan	Menunjukkan simpanan data *non computer*/informasi berupa *file* pada proses manual. (Dokumen yang disimpan pada lemari, arsip, map, dll).
	Penyimpanan magnetik	Media penyimpanan yang dilakukan untuk proses terkomputerisasi.
	Kondisi	Keputusan menunjukkan pilihan iya atau tidak.
	Arsip digital	Menunjukkan simpanan data terkomputerisasi.
	*Display*	Menunjukkan output yang ditampilkan di monitor.

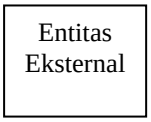
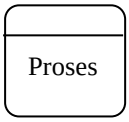
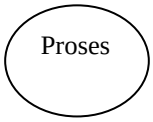
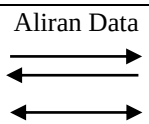
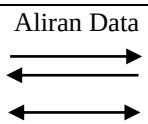
	Dokumen	Menunjukkan dokumen *input* dan *output* baik untuk proses manual atau komputer (melalui printer).

2.10 *Data Flow Diagram* (DFD)

Data Flow Diagram adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan ke mana tujuan data yang keluar dari sistem, di mana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut. DFD menggambarkan penyimpanan data dan proses yang mentransformasikan data. DFD menunjukkan hubungan antara data pada sistem dan proses pada sistem (Kristanto, 2022, p. 61).

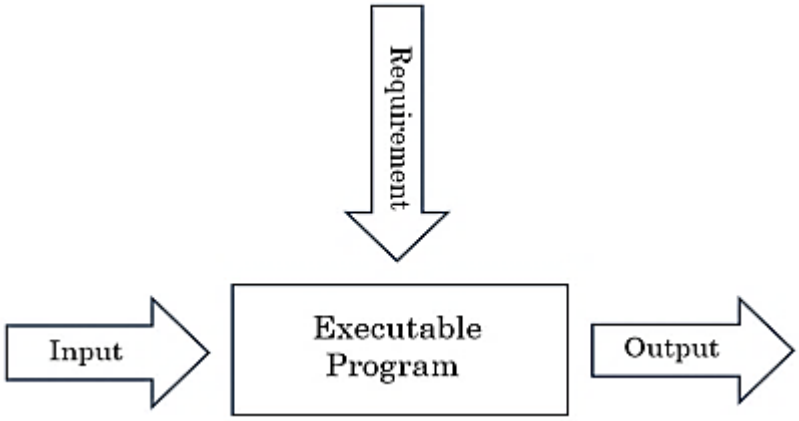
Ada beberapa simbol DFD yang dipakai untuk menggambarkan data beserta proses transformasi data, antara lain :

Tabel 2.13 Simbol DFD

Gane/Sarson	**Yourdon/De Marco**	**Keterangan**
		Entitas eksternal, dapat berupa orang atau unit terkait yang berinteraksi dengan sistem, tetapi di luar sistem.
		Orang atau unit yang menggunakan atau melakukan transformasi data, tanpa mengidentifikasi komponen fisik.
		Aliran data dengan arah tertentu dari sumber ke tujuan.
		Penyimpanan data atau tempat data, di mana data dapat dilihat oleh proses.

2.11 Pengujian *Black Box*

Pengujian *black box* adalah pengujian yang berfokus pada pengujian persyaratan fungsional perangkat lunak untuk memastikan bahwa fungsi-fungsi yang ada berfungsi dengan baik, tanpa memperhatikan struktur logika internal perangkat lunak. Tujuannya untuk memastikan bahwa program dapat dijalankan tanpa kesalahan dengan menguji berbagai transaksi (*input, edit, delete, update* data) serta menghasilkan *output* sesuai kebutuhan. Pengujian *black box* digunakan untuk menemukan kesalahan dalam fungsi-fungsi yang tidak benar atau hilang, kesalahan *interface*, struktur data, akses ke *database* eksternal, kesalahan dalam alur kerja, serta masalah dalam inisiasi dan terminasi. Pengujian *black box* dapat direpresentasikan sebagai berikut (Dawis et al., 2023) :



Gambar 2.4 Representasi *Black Box Testing*