BAB II

LANDASAN TEORI

2.1 Peramalan (*Forecasting*)

2.1.1 Peramalan

Peramalan adalah metode untuk memperkirakan suatu nilai dimasa depan dengan menggunakan data masa lalu. Peramalan yang baik adalah peramalan yang dilakukan dengan mengikuti langkah – langkah atau prosedur penyusunan yang baik akan menentukan kualitas atau mutu dari hasil peramalan yang disusun (Wardah & Iskandar, 2016). Ada 3 langkah peramalan yang penting, yaitu:

1. Menganalisa data yang lalu, tahap ini berguna untuk pola yang terjadi pada masa lalu.
2. Menentukan data yang dipergunakan. Metode yang baik adalah metode yang memberikan hasil ramalan yang tidak jauh berbeda dengan kenyataan yang terjadi.
3. Memproyeksikan data yang lalu dengan menggunakan metode yang dipergunakan, dan mempertimbangkan adanya beberapa faktor perubahan (perubahan kebijakan – kebijakan yang mungkin terjadi, termasuk perubahan kebijakan pemerintah, perkembangan potensi masyarakat, perkembangan teknologi dan penemuan – penemuan baru).

2.1.2 Prinsip Peramalan

Menurut (Wardah & Iskandar, 2016), prinsip – prinsip peramalan yang perlu dipertimbangkan adalah sebagai berikut:

1. Peramalan melibatkan kesalahan *(error)*, peramalan akan hanya mengurangi ketidakpastian tetapi tidak menghilangkan.
2. Peramalan sebaiknya memakai tolak ukur kesalahan peramalan, pemakai harus tahu besar kesalahan, yang dapat dinyatakan dalam satuan unit atau persentase *(probability)* permintaan aktual akan jatuh dalam interval peramalan.
3. Peramalan jangka pendek lebih akurat daripada peramalan jangka panjang, karena peramalan jangka pendek yaitu kondisi yang mempengaruhi permintaan cenderung tetap atau berubah lambat, sehingga peramalan jangka pendek lebih akurat.

2.1.3 Karakteristik Peramalan

Menurut (Wardah & Iskandar, 2016), karakteristik peramalan yang baik adalah sebagai berikut:

1. *Accuracy* (Ketepatan).
2. *Low cost of software purchase or development* (Biaya pembelian software atau pengembangan yang rendah).
3. *Low computer time requirements* (Persyaratan waktu computer yang rendah).
4. *Low computer storage requirements* (Persyaratan penyimpanan komputer yang rendah).

2.2 Metode Sistem

2.2.1 Metode *Weighted Moving Average*

Metode *Weighted Moving Average* merupakan metode yang cocok digunakan pada data yang bersifat *time-series*, yaitu data yang berubah dari waktu ke waktu.

Di dalam metode *Weighted Moving Average*, selain perhitungannya sederhana, pada teknik *Weighted Moving Average* diberikan bobot yang berbeda untuk setiap data historis masa lalu yang tersedia, dengan asumsi bahwa data historis yang paling terakhir atau terbaru akan memiliki bobot lebih besar dibandingkan dengan data historis yang lama karena data yang paling terakhir atau terbaru merupakan data yang paling relevan untuk peramalan. Keunggulan lainnya dari metode ini adalah pemberian nilai bobotnya dapat disesuaikan, tetapi penentuan bobot optimalnya sulit (Riyanto et al., 2017).

Secara matematis perhitungan *Weighted Moving Average* dirumuskan dalam persamaan berikut:

$$WMA = \frac{\sum(data \times bobot)}{\sum bobot}$$

Keterangan:

Data = Data actual pada periode t

Bobot = Penilaian sesuai panjang periode

2.2.2 Metode *Double Exponential Smoothing*

Metode *Double Exponential Smoothing* adalah suatu metode yang paling luas digunakan untuk menentukan persamaan trend data pemulusan kedua melalui proses *smoothing*. *Double Exponential Smoothing* digunakan untuk data yang memiliki trend atau data yang memiliki kecenderungan peningkatan atau penurunan dalam jangka panjang. Sistem peramalan ini menangkap pola dari data yang telah lalu kemudian digunakan untuk memproyeksikan data yang akan datang.

Metode *Double Exponential Smoothing* merupakan model linear yang dikemukakan oleh *Brown*. Dalam metode ini dilakukan proses *smoothing* dua kali. Dasar pemikiran metode pemulusan eksponensial linear dari *Brown* adalah serupa dengan rata-rata bergerak linear, karena kedua nilai pemulusan tunggal dan ganda ketinggalan dari data yang sebenarnya jika terdapat unsur trend. Perbedaan antara nilai pemulusan tunggal dan ganda dapat ditambahkan dengan nilai pemulusan tunggal dan disesuaikan untuk trend (Lieberty & Imbar, 2015). Persamaan yang dipakai dalam implementasi pemulusan eksponensial linear satu-parameter dari *Brown* adalah sebagai berikut:

1. Pemulusan Eksponensial Tunggal : $S'_t = \alpha X_t + (1 - a)\, S'_{t-1}$
2. Pemulusan Eksponensial Ganda : $S''_t = \alpha S'_t + (1 - a)\, S''_{t-1}$
3. Pemulusan Trend : $\alpha_t = 2S'_t - S''_t$

$$b_t = \frac{\alpha}{1 - \alpha}\,(S'_t - S''_t)$$

4. Ramalan : $F_{t+m} = \alpha_t + b_t \,.m$

Keterangan:

S'_t = Nilai pemulusan tunggal

S''_t = Nilai pemulusan ganda

Xt = Data sebenarnya pada waktu ke t

m = Jumlah periode bulan yang akan diramalkan

2.3 Metode Mengukur *Accuracy* dan *Error* dalam *Forecasting*

2.3.1 *Mean Absolute Deviation* (*MAD*)

Mean Absolute Deviation (MAD) mengukur ketepatan ramalan dengan rata – rata kesalahan kesalahan dugaan (nilai absolut masing – masing kesalahan). MAD berguna ketika mengukur kesalahan ramalan dalam unit yang sama sebagai deret asli (Riyanto et al., 2017). Nilai MAD dapat dihitung dengan menggunakan rumus sebagai berikut:

$$MAD = \frac{\sum_{t=1}^{n} |y(t) - y'(t)|}{n}$$

Keterangan:

y(t) = Nilai aktual pada periode t

y'(t) = Nilai hasil peramalan pada periode t

t = Periode peramalan

n = Banyaknya data

2.3.2 *Mean Squared Error* (*MSE*)

Mean Squared Error adalah kuadrat rata – rata kesalahan meramal (Lieberty & Imbar, 2015). Rumus yang dipakai untuk menghitung MSE adalah sebagai berikut:

$$MSE = \frac{\sum (X_t - F_t)^2}{n}$$

Metode ini mudah menghitungnya dan sederhana, tetapi mempunyai kelemahan – kelemahan antara lain:

1. Memerlukan data historis yang cukup.
2. Data tiap periode diberi *weight* (bobot) sama.

3. Jika fluktuasi data tidak random, tidak menghasilkan *forecasting* yang baik.

2.3.3 *Mean Absolute Percentage Error* (*MAPE*)

Percentage Error merupakan kesalahan persentase dari suatu peramalan, dimana:

$$PE = \left(\frac{X_t - F_t}{X_t}\right) . 100$$

Mean Absolute Percentage Error merupakan nilai tengah kesalahan persentase *absolute* dari suatu peramalan.

$$MAPE = \frac{\sum |PE|}{n}$$

Semakin kecil nilai MAPE berarti nilai taksiran semakin mendekati nilai sebenarnya, atau metode yang dipilih merupakan metode terbaik (Lieberty & Imbar, 2015).

2.4 Pemrograman Aplikasi

2.4.1 PHP (*Hypertext Preprocessor*)

PHP merupakan salah satu bahasa pemrograman yang berjalan dalam sebuah web server dan berfungsi sebagai pengolah data pada sebuah server. PHP bekerja didalam sebuah dokumen HTML *(Hypertext Markup Language)* untuk dapat menghasilkan isi dari sebuah halaman web sesuai permintaan. Dengan PHP, kita dapat merubah situs kita menjadi sebuah aplikasi berbasis web, tidak lagi hanya sekedar sekumpulan halaman statistik, yang jarang diperbaharui.

Untuk membuat website yang dinamis dan mudah di *update* setiap saat dari browser, dibutuhkan sebuah program yang mampu mengolah data dari komputer

client atau dari komputer server itu sendiri sehingga mudah dan nyaman disajikan di browser (Mubarak, 2019).

2.4.2 *JavaScript*

JavaScript merupakan bahasa scripting yang pada awalnya dikembangkan oleh *Netscape*. Dengan menggunakan *javascript*, dapat dibuat tampilan web yang lebih interaktif lagi. Agar user dapat menjalankan *javascript*, alat yang dibutuhkan hanyalah *browser* yang kemampuan *javascript*-nya telah diaktifkan. Penulisan *javascript* dapat dipadukan pada halaman HTML, dan dari sini dapat dilihat bahwa *javascript* merupakan bahasa pemrograman yang berjalan di sisi *client*. Karena proses dari *javascript* tergantung proses dari *browser* yang digunakan oleh *user*.

JavaScript adalah bahasa pemrograman berbentuk kumpulan *script* yang berjalan pada suatu dokumen HTML. *JavaScript* dapat menyempurnakan tampilan dan sistem pada halaman web yang dikembangkan (Mariko, 2019).

2.4.3 *Database*

Database merupakan kumpulan file – file yang saling berkaitan dan berinteraksi, relasi tersebut bila ditunjukan dengan kunci dari tiap – tiap file yang ada. Satu *database* menunjukkan suatu kumpulan data yang dipakai dalam suatu lingkup perusahaan, instansi. Pengolahan *database* merupakan suatu cara yang dilakukan terhadap file – file yang berada di suatu instansi yang mana file tersebut dapat disusun, diurut, diambil sewaktu – waktu serta dapat ditampilkan dalam bentuk suatu laporan sehingga dapat mengolah file –file yang berisikan informasi tersebut secara teratur. Suatu *database* dibentuk untuk mengatasi masalah yang

sering dihadapi di dalam pengolahan data seperti: keamanan data, kesulitan mengakses data (Sovia & Febio, 2011).

2.5 *Unified Modeling Language* (UML)

UML adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO *(Object-Oriented)*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas – kelas dalam bahasa program yang spesifik, skema database, dan komponen – komponen yang diperlukan dalam sistem software (Mubarak, 2019).

2.5.1 *Use Case Diagram*

Use Case Diagram merupakan gambaran dari fungsionalitas yang diharapkan dari sebuah sistem, dan mempresentasikan sebuah interaksi antara aktor dan sistem. Didalam *use case* terdapat aktor yang merupakan sebuah gambaran entitas dari manusia atau sebuah sistem yang melakukan pekerjaan di sistem (Prihandoyo, 2018). Berikut adalah simbol – simbol yang ada pada *use case diagram*:

Table 1. 1 *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri*(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Sumber : (Hasanudin, 2021)

2.5.2 *Activity Diagram*

Activity Diagram merupakan gambaran alir dari aktivitas – aktivitas didalam sistem yang berjalan (Prihandoyo, 2018). Berikut adalah simbol – simbol yang ada pada *activity diagram*:

Table 1. 2 *Activity Diagram*

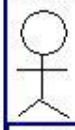
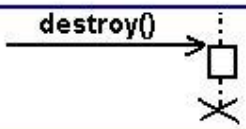
Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem,aktivitas biasanya diawali dengan kata kerja.
	Percabangan / Decision	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan / Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	Swimlane	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : (Hasanudin, 2021)

2.5.3 *Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar objek didalam dan di sekitar sistem yang berupa *message* yang digambarkan terhadap waktu (Prihandoyo, 2018). Berikut adalah simbol – simbol yang ada pada *sequence diagram*:

Table 1. 3 *Sequence Diagram*

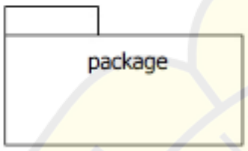
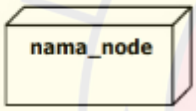
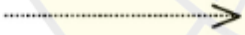
aktor  atau nama_aktor	• orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari system. • Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan. • Ditempatkan di bagian atas diagram.
objek objek:kelas	**Sebuah objek:** • **Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan.** • **Ditempatkan di bagian atas diagram.**
Garis hidup objek 	• Menandakan kehidupan obyek selama urutan. • diakhiri tanda X pada titik di mana kelas tidak lagi berinteraksi.
Objek sedang aktif berinteraksi 	**Fokus kontrol:** • **Adalah persegi panjang yang sempit panjang ditempatkan di atas sebuah garis hidup.** • **Menandakan ketika suatu objek mengirim atau menerima pesan.**
pesan pesan() 	objek mengirim satu pesan ke objek lainya
<<create>> 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
1:masukan 	menyatakan bahwa suatu objek mengirimkan masukan ke objek lainnya arah panah mengarah pada objek yang dikirimi
1:keluaran 	**objek/metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian**
	**menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy**

Sumber : (Prihandoyo, 2018)

2.5.4 *Deployment Diagram*

Deployment Diagram menggambarkan hubungan antara *software* dan *hardware* terhadap sistem dan apa saja *output* yang dihasilkan (Irmayani & Susyatih, 2017). Berikut adalah simbol – simbol yang ada pada *deployment diagram*:

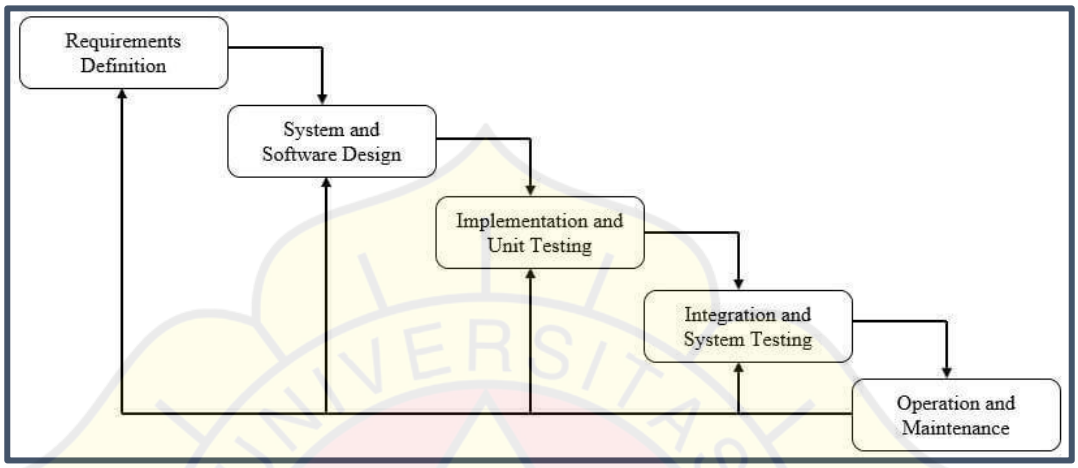
Table 1. 4 *Deployment Diagram*

Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih *node*
Node 	biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / *dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
Link	relasi antar *node*

Sumber : (Irmayani & Susyatih, 2017)

2.6 Metode Pengembangan Sistem

Metode *waterfall* merupakan model pengembangan sistem informasi yang sistematik dan sekuensial (Sasmito, 2017). Metode *waterfall* memiliki tahapan – tahapan sebagai berikut:



Sumber : (Sasmito, 2017)

Gambar 2. 1 *Waterfall*

1. *Requirements definition*

   Layanan sistem, kendala, dan tujuan ditetapkan oleh hasil konsultasi dengan pengguna yang kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.

2. *System and software design*

   Tahapan perancangan sistem mengalokasikan kebutuhan – kebutuhan sistem baik perangkat keras maupun perangkat lunak dengan membentuk arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksi sistem dasar perangkat lunak dan hubungannya.

3. *Implementation and unit testing*

Perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.

4. *Integration and system testing*

   Unit – unit individu program atau program digabung dan diuji sebagai sebuah sistem lengkap untuk memastikan apakah sesuai dengan kebutuhan perangkat lunak atau tidak. Setelah pengujian, perangkat lunak dapat dikirimkan ke *customer*.

5. *Operation and maintenance*

   Sistem dipasang dan digunakan secara nyata. *Maintenance* melibatkan pembetulan kesalahan yang tidak ditemukan pada tahapan – tahapan sebelumnya, meningkatkan implementasi dari unit sistem, dan meningkatkan layanan sistem sebagai kebutuhan baru.

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA