BAB II

LANDASAN TEORI

2.1. Peramalan (Forecasting)

Peramalan (*forecasting*) sangat penting bagi setiap organisasi, karena hal ini akan menjadi dasar pengambilan keputusan manajemen yang tentunya akan mempengaruhi perkembangan organisasi. Metode peramalan berfungsi untuk memprediksi data runtut waktu (*time series*) beberapa periode yang akan datang berdasarkan data beberapa periode sebelumnya. Peramalan adalah memprediksi kejadian yang akan datang dengan menggunakan data-data sebelumnya (data historis) yang nantinya akan menjadi dasar pengambilan keputusan. (Makridakis, 2000) Untuk melakukan peramalan diperlukan metode tertentu dan metode mana yang digunakan tergantung dari data dan informasi yang akan diramal serta tujuan yang hendak dicapai. Dalam prakteknya terdapat berbagai metode peramalan antara lain (Herjanto, 2008)

1. Peramalan berdasarkan jangka waktu.

   Peramalan berdasarkan jangak waktu dibagi menjadi 3 yaitu

   a. *Peramalan jangka pendek* . Jangaka waktu untuk peramalan kurang satu tahun, umumnya kurang tiga bulan, biasanya digunakan untuk rencana pembelian, penjadwalan kerja, jumlah TK, tingkat produksi
   b. Peramalan jangka menengah. Jangka waktu untuk peramalan tiga bulan hingga 18 bulan, biasanya digunakan untuk perencanaan

penjualan, perencanaan dan penganggaran produksi dan menganalisis berbagai rencana operasi

c. Peramalan jangka panjang. Jangka waktu untuk peramalan tiga tahun atau lebih, biasanya digunakan untuk merencanakan produk baru, penganggaran modal, lokasi fasilitas, atau ekspansi dan penelitian serta pengembangan.

2. Peramalan berdasarkan metode / pendekatan

Peramalan berdasarkan metode terbagi menjadi 2 yaitu

a. Metode peramalan kualitatif. Permalan kualitatif adalah peramalan yang didasarkan atas data kualitatif pada masa lalu. Hasil peramalan yang dibuat sangat bergantung pada orang yang menyusunnya. Metode kualitatif dapat dibagi menjadi metode eksplanatoris dan normatif

b. Metode peramalan kuantitatif. Permalan kuantitatif adala peramalan yang didasarkan atas data kuantitatif pada masa yang lalu. Hasil peramalan yang dibuat sangat bergantung pada metode yang dipergunakan dalam peramalan tersebut. Metode kuantitatif dibagi menjadi 2 yaitu deret berkala (Time Series) dan metode kausal.

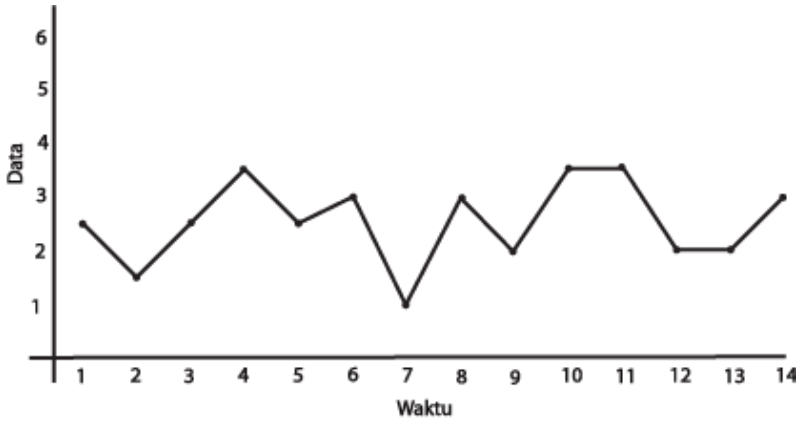
2.2. Data Runtun Waktu (*Time Series*)

Peramalan didasarkan pada data historis atau data runtut waktu sehinga memperoleh nilai dugaan dari suatu periode tertentu. Runtut waktu (*time series*) adalah hasil pengamatan kontinyu atau data yang disusun berdasarkan urutan waktu (Boedijoewono, 2016)

Sebuah runtut waktu (*time series*) memiliki beberapa pola. Pola-pola tersebut akan terlihat pada runtut waktu yang sudah digambarkan melalui grafik. Pola-pola tersebut dapat dijelaskan dengan banyaknya kemungkinan hubungan sebab akibat. Adapun pola-pola umum runtut waktu adalah sebagai berikut (Assuri, 2016)

1. Pola Acak (*Random*)

   Pola yang muncul karena peristiwa yang tidak terduga seperti perang sehingga menghasilkan pola yang tidak beraturan. Seperti pada Gambar 2.1 pola acak, fluktuasi data berubah-ubah tidak dapat diprediksi.



Gambar 2.1 Bentuk Umum Pola Acak

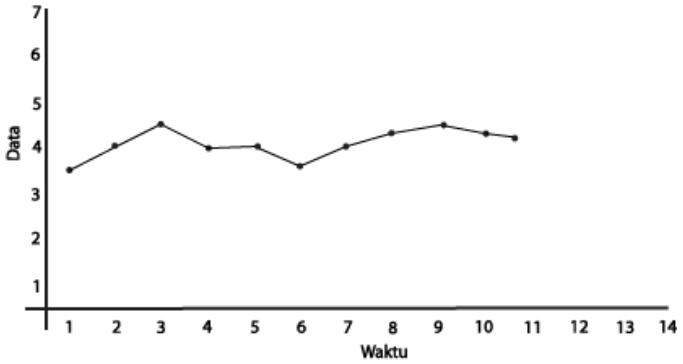
2. Pola Musiman

Pola musiman dihasilkan oleh kejadian yang terjadi secara musiman atau periodik contoh: iklim, hari besar keagamaan, kwartal waktu terntu. Seperti pada gambar 2.2 terjadi peningkatan diperiode 6, 12, dan 18.

Gambar 2.2 Bentuk Umum Pola Musiman

3. Pola Siklus

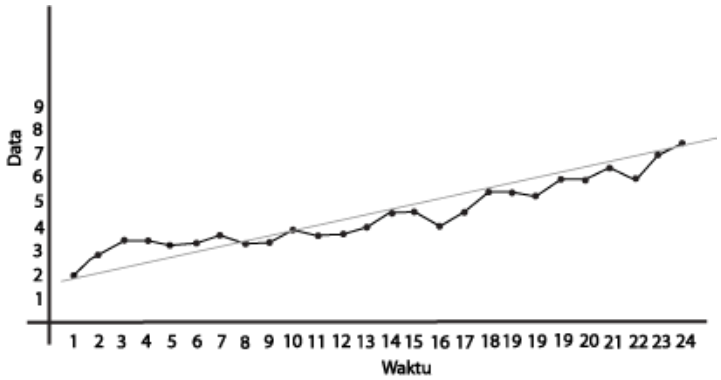
Pola siklus adalah pola perubahan naik atau turun, sehingga siklus ini berubah dan bervariasi dari suatu siklus ke siklus berikutnya. Pola siklus dihasilkan oleh pengaruh ekspansi ekonomi.



Gambar 2.3 Bentuk Umum Pola Siklus

4. Pola *Trend*

Pola *trend* disebabkan oleh perubahan jangka panjang yang terjadi di sekitar faktor-faktor yang mempengaruhi data runtut waktu (*time series*). Di dalam pola *trend* terjadi peningkatan atau penurunan selama beberapa periode tertentu. Seperti pada gambar 2.4 pola data mempunyai kencenderungan meningkat.



Gambar 2.4 Bentuk Umum Pola Trend

2.3. Metode *Exponential Smoothing*

Exponential Smoothing adalah suatu prosedur yang secara terus menerus memperbaiki peramalan dengan merata-rata nilai masa lalu dari suatu data runtut waktu dengan cara menurun (exponential). Macam-macam metode *exponential smoothing* yaitu (Makridakis, 2000)

2.3.1. *Single Exponential Smoothing*

Metode *Single Exponential Smoothing* menambahkan parameter *alpha* dalam model untuk mengurangi faktor kerandoman. Nilai prediksi dapat dicari dengan menggunakan rumus berikut (Makridakis, 2000)

$$\hat{Y}_{t+1} = \alpha Y_t + (1-\alpha)\hat{Y}_t \quad (1)$$

dengan

$\hat{Y}_{t+1}$: nilai prediksi untuk periode berikutnya

α : konstanta *smoothing*

Y_t : data sebenarnya pada periode *t*

$\hat{Y}_t$: nilai prediksi pada periode t yang diperoleh dari rata-rata penghalusan hingga periode t – 1

2.3.2. *Double Exponential Smoothing*

Metode *single exponential smoothing* hanya akan efektif apabila data *time series* yang diamati memiliki pola horizontal. Jika metode itu digunakan untuk data *time series* yang memiliki unsur *trend* yang konsisten, nilai-nilai perkiraan akan selalu berada di belakang nilai aktualnya.

2.3.2.1. Metode Brown's

Metode ini dikembangkan oleh Brown's untuk mengatasi perbedaan yang muncul antara data aktual dan nilai peramalan apabila ada trend pada polanya. Persamaan yang digunakan pada metode ini adalah (Makridakis, 2000)

$$A_t = \alpha Y_t + (1-\alpha)A_{t-1} \quad (2)$$

$$A'_t = \alpha A_t + (1-\alpha)A'_{t-1} \quad (3)$$

$$a_t = 2A_t - A'_t \quad (4)$$

$$b_t = \frac{\alpha}{1-\alpha}(A_t - A'_t) \quad (5)$$

Persamaan yang digunakan untuk membuat peramalan pada periode *p* yang akan datang adalah:

$$\hat{Y}_{t+p} = a_t + b_t p \quad (6)$$

dengan

A_t : nilai *exponential smoothing*

A'_t : nilai double *exponential smoothing*

α : konstanta pemulusan

a_t : perbedaan antara nilai-nilai *exponential smoothing*

b_t : estimasi *trend*

Y_t : nilai aktual pada periode *t*

p : jumlah periode ke depan yang akan diramalkan

2.3.2.2. Metode Holt

Metode ini nilai trend tidak dimuluskan dengan pemulusan ganda secara langsung, tetapi proses pemulusan trend dilakuakan dengan parameter berbeda dengan parameter pada pemulusan data asli. Secara matematis metode ini ditulis pada tiga persamaan (Makridakis, 2000)

$$A_t = \alpha Y_t + (1-\alpha)(A_{t-1} + T_{t-1}) \quad (7)$$

$$T_t = \beta(A_t - A_{t-1}) + (1-\beta)T_{t-1} \quad (8)$$

Persamaan yang digunakan untuk membuat peramalan pada periode p yang akan datang adalah:

$$\hat{Y}_{t+p} = A_t + T_t p \qquad (9)$$

dengan

A_t : nilai *exponential smoothing*

α : konstanta pemulusan untuk data ($0 \leq \alpha \leq 1$)

β : konstanta pemulusan untuk estimasi trend ($0 \leq \beta \leq 1$)

Y_t : nilai aktual pada periode t

T_t : estimasi *trend*

p : jumlah periode ke depan yang akan diramalkan

Untuk menentukan besarnya nilai α untuk data, kita dapat melihat pola historis data aktualnya. Apabila pola historis sangat bergejolak atau tidak stabil dari waktu ke waktu maka kita memilih nilai α mendekati 1. Sebaliknya apabila pola historis relatif stabil kita dapat memilih nilai α mendekati 0, begitu juga sama dengan menentukan nilai β Apabila pola data estimasi trend tidak stabil maka kita memilih nilai β mendekati 1. Sebaliknya apabila pola data stabil maka kita memilih nilai β mendekati 0.

2.4. *Unified Modelling Language* (UML)

Pemodelan perangkat lunak dalam pembangunan perangkat lunak sangatlah penting, karena pemodelan berfungsi untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem informasi sehingga lebih terencana. Perangkat pemodelan adalah suatu model yang digunakan untuk menguraikan sistem menjadi bagian-bagian yang dapat diatur dan mennkomunikasikan ciri konseptual dan fungsional kepada pengamat. Salah satu perangkat pemodelan adalah *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasi, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. (Salahuddin & S, 2013)

Pada UML 2.3 terdiri dari 13 macam yang dibagi menjadi 3 kategori yaitu :

1. *Structure diagrams* adalah kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* adalah kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem
3. *Interaction diagrams* adalah kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu system

2.4.1. Diagram Kelas (*Class Diagram*)

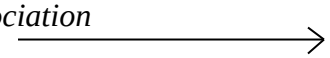
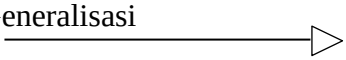
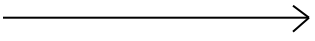
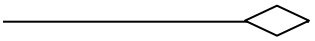
Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Setiap kelas memiliki atribut dan metode atau operasi. Atribut adalah variabel-variabel yang dimiliki oleh suatu kelas. Metode atau operasi adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis- jenis kelas berikut :

1. Kelas main. Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*). Kelas yang mendifinisikan dan mengatur tampilan-tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case (controller)*. Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data atau model. Kelas yang digunakan untuk membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Berikut adalah simbol–simbol yang ada pada diagram kelas (Salahuddin & S, 2013)

Tabel 2.1 Simbol Diagram Kelas

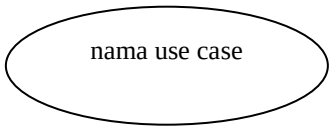
Simbol	Deskripsi
Kelas nama_kelas +atribut +operasi()	Kelas pada struktur system
Antar muka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesifikasi (umum khusus)
Kebergantungan / *dependency* 	Kebergantungan antar kelas
Agregrasi / *aggregation* 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

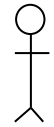
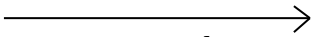
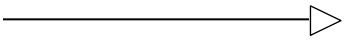
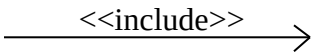
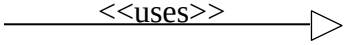
2.4.2. Diagram *Use Case* (*Use Case Diagram*)

Diagram *use case* adalah pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara garis besar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuiah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Ada dua hal utama pada *use case* yaitu pendefinisian actor dan *use case*. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor.

Berikut ini adalah simbol-simbol yang ada pada diagram *use case* (Salahuddin & S, 2013)

Tabel 2.2 Simbol Diagram *Use Case*

Simbol	Deskripsi
Use case 	Fungsionalitas yang akan disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*
Aktor / *actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem

Simbol	Deskripsi
 nama aktor	informasi yang akan dibuat itu sendiri, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association 	Komunikasi antara aktor dan *use case* yang berpartisasi pada *use case* atau *use case* memiliki interkasi dengan aktor
Ekstensi / extend  <<extend>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu
Generalisasi / generalization 	Hubungan generalisasi dan spesisialisasi (umum-khusus) antara dua buah use case diamana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
Menggunakan / include / uses  <<include>>  <<uses>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini

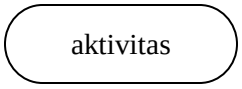
2.4.3. Diagram Aktivitas (*Activity Diagram*)

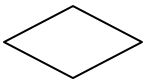
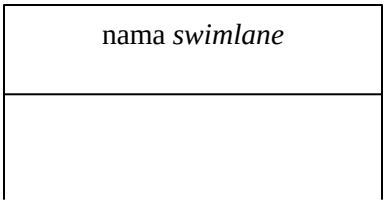
Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis adalah setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dengan setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3. Rancangan pengujian adalah setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4. Rancangan menu yang ditampilkan pada perangkat lunak

Berikut adalah simbol-simbol yang ada diagram aktivitas (Salahuddin & S, 2013)

Tabel 2.3 Simbol Diagram Aktivitas

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem
Percabangan / *decision*	Asosiasi percabangan dimana jika

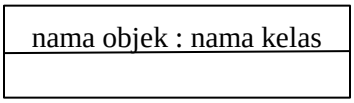
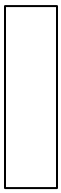
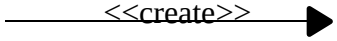
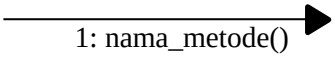
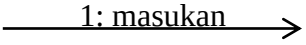
Simbol	Deskripsi
	ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhhir 	Status akhir yang dilakukan sistem, sebiah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap akktivitas yang terjadi

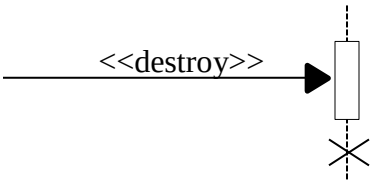
2.4.4. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Berikut adalah simbol-simbol yang ada pada diagram sekuen. (Salahuddin & S, 2013)

Tabel 2.4 Simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri,

Simbol	Deskripsi
	biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / lifeline	Menyatakan kehidupan sutau objek
Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya
Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yangdibuat
Pesan tipe call 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode
Pesan tipe send 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return	Menyatakan bahwa suatu objek yang telah menjalankan suatu

Simbol	Deskripsi
1 : keluaran	operasi atau metode menghasilkan suatu kembalian ke objek terntentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	Menyatakan bahwa suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhir

2.5. VB.Net

Microsoft Visual Basic .NET adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem .NET Framework, dengan menggunakan bahasa BASIC. Dengan menggunakan alat ini, para *programmer* dapat membangun aplikasi Windows Forms, Aplikasi web berbasis ASP.NET, dan juga aplikasi *command-line*. Alat ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti Microsoft Visual C++, Visual C#, atau Visual J#), atau juga dapat diperoleh secara terpadu dalam Microsoft Visual Studio .NET. Bahasa Visual Basic .NET sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari Microsoft Visual Basic versi sebelumnya yang diimplementasikan di atas .NET Framework. Peluncurannya mengundang kontroversi, mengingat banyak sekali perubahan yang dilakukan oleh Microsoft, dan versi baru ini tidak kompatibel dengan versi terdahulu.

2.6. SQL Server 2008

SQL Server memakai sebuah tipe database yang dinamakan database reasional. Database relasional adalah database mengorganisasikan data dalam bentuk tabel. Tabel dibentuk dengan mengelompokan data yang mempunyai subjek yang sama. Tabel berisi baris dan kolom informasi. Tabel-tabel dapat saling berhubungan jika dihubungkan. SQL Server 2008 merupakan bahasa pemrograman yang dirancang khusus untuk berkomunikasi dengan database relasional guna mendukung aplikasi dengan arsitektur client/server Authentication windows.

Bibliography

Assuri, S. (2016). *Manajemen Operasi Produksi Pencapaian Sasaran Organisasi Berkesinambungan.* Jakarta: Rajagrafindo Persada.

Boedijoewono, N. (2016). *Pengantar Statistika Ekonomi dan Bisnis.* Jogjakarta: UPP STIM YKPN.

Herjanto, E. (2008). *Manajemen Operasi Edisi Ketiga.* Jakarta: Grasindo.

Makridakis, S. (2000). *Metode dan Aplikasi Permalan Jilid 2.* Jakarta: Interakasara.

S, R. A., & Salahuddin, M. (2013). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* Bandung: Informatika Bandung.