BAB II

LANDASAN TEORI

2.1. Aplikasi

Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan-pekerjaan tertentu (khusus), salah satunya adalah aplikasi file text. Aplikasi file text merupakan sebuah program yang dapat menyimpan text atau tulisan dalam extention .txt. Untuk membuat aplikasi ini dibutuhkan komponen label, textbox, dan command. (Hendrayudi, 2009)

2.2. Perencanaan Persediaan

2.2.1. Pengertian Perencanaan Persediaan

Perencanaan merupakan suatu proses yang melibatkan penentuan sasaran atau tujuan organisasi, sedangkan persediaan adalah sumber daya yang menggangur (*idle resources*) yang menunggu proses lebih lanjut. Proses lebih lanjut tersebut adalah berupa kegiatan produksi pada sistem manufaktur, kegiatan pemasaran pada sistem distribusi atau kegiatan konsumsi panggan pada rumah tangga. Persediaan (*inventory*) didefinisikan sebagai sumber daya yang disimpan untuk memenuhi permintaan saat ini maupun yang akan datang.

Perencanaan persediaan merupakan salah satu kegiatan pengawasan persediaan yang menjadi dasar untuk segala tindakan pengawasan persediaan yang akan dijalankan. Dalam perencanaan persediaan ditentukan usaha-usaha atau tindakan yang akan atau perlu diambil untuk mencapai hasil yang sesuai dengan rencana, dengan mempertimbangkan masalah-masalah yang mungkin timbul di

masa yang akan datang, seperti mempertimbangkan jumlah kapasitas maksimal penyimpanan, jumlah stok saat ini yang tersedia dan jumlah kebutuhan dimasa yang akan datang. Dengan kata lain sebuah perencanaan persediaan dihasilkan berdasarkan hasil selisih antara jumlah kebutuhan untuk masa yang akan datang dengan jumlah stok saat ini, serta harus mempertimbangkannya hasilnya dengan total kapasitas maksimal dari penyimpanannya.

Dalam penerapanya, kegiatan perencanaan persediaan tidak dapat hanya berdasarkan sebuah perhitungan rumus, tetapi terdapat aturan-aturan yang ditetapkan dalam kegiatan tersebut. Apabila sebuah hasil dari perencanaan persediaan melebihi aturan yang telah ditetapkan oleh perusahaan dalam perencanaan persediaan, maka perencanaan tersebut akan mengikuti aturan tersebut. (Nasution dan Prasetyawan, 2008)

2.2.2. Jenis-Jenis Persediaan

Persediaan bisa diklasifikasikan dengan berbagai cara. Pada bagian ini menjelaskan mengenai persediaan berdasarkan bentuknya adalah sebagai berikut:

a. Persediaan bahan mentah (*raw material inventory*) merupakan persediaan yang digunakan untuk melakukan *decouple* (memisahkan) pemasok dari proses produksi.
b. Persediaan barang setengah jadi (*work in process* – WIP *inventory*) adalah komponen-komponen atau bahan mentah yang telah melewati beberapa proses perubahan, tetapi belum selesai. WIP ada karena waktu yang diperlukan untuk menyelesaikan sebuah produk.

c. MRO (*Maintenance Repair Operation*) adalah persediaan-persediaan yang disediakan untuk persediaan pemeliharaan, perbaikan, operasi yang dibutuhkan untuk menjaga agar mesin-mesin dan proses-proses tetap produktif.
d. Persediaan barang jadi adalah produk yang telah selesai dan tinggal menunggu pengiriman. Barang jadi yang dapat dimasukkan ke persediaan karena permintaan pelanggan di masa mendatang tidak diketahui. (Heizer & Render, 2010)

2.3. *Businsess Analysis Body of Knowledge* (BABOK)

Business analysis adalah himpunan tugas dan teknik yang digunakan untuk bekerja sebagai penghubung antara para pemangku kepentingan untuk memahami struktur, kebijakan, dan operasi dari organisasi, dan untuk merekomendasikan solusi yang memungkinkan organisasi untuk mencapai tujuan. Dalam melakukan *business analysis* terdapat beberapa klasifikasi dalam *requirements* antara lain sebagai berikut: (Simues, 2009)

2.3.1. *Business Requirement*

Business requirement adalah kebutuhan tingkat yang lebih tinggi dari tujuan, sasaran, atau kebutuhan perusahaan yang menjelaskan alasan mengapa proyek telah dimulai, tujuan bahwa proyek akan dicapai, dan *matrix* yang digunakan untuk mengukur keberhasilannya.

2.3.2. *Stakeholder Requirement*

Stakeholder management adalah pernyataan dari kebutuhan *stakeholder* tertentu atau kelas *stakeholder* yang menggambarkan kebutuhan dari pemangku kepentingan dan bagaimana pemangku kepentingan akan berinteraksi dengan solusi.

2.3.3. *Solution Requirement*

Solutions requirement merupakan gambaran karakteristik dari solusi yang memenuhi kebutuhan bisnis dan kebutuhan *stakeholder*. *Solutions requirement* dibagi menjadi dua sub-kategori dalam menggambarkan solusi perangkat lunak yaitu:

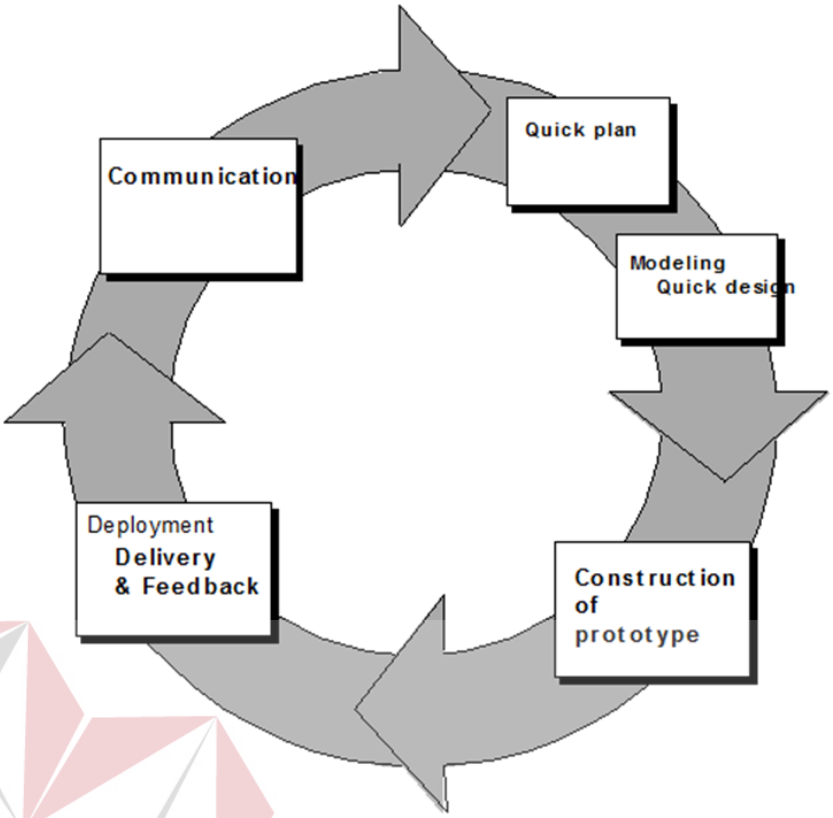
a. *Functional requirement* yaitu menggambarkan perilaku dan informasi bahwa solusi akan dikelola. Mereka menggambarkan kemampuan sistem akan dapat melakukan dalam hal perilaku atau operasi khusus tindakan aplikasi teknologi informasi atau tanggapan.
b. *Non-functional requirement* yaitu menangkap kondisi yang tidak langsung berhubungan dengan perilaku atau fungsi dari solusi, melainkan menggambarkan kondisi lingkungan di mana solusi harus tetap efektif atau kualitas bahwa sistem harus memiliki. Mereka juga dikenal sebagai kualitas atau kebutuhan tambahan. Ini dapat mencakup kebutuhan yang terkait dengan kapasitas, kecepatan, keamanan, ketersediaan dan arsitektur informasi dan presentasi dari *user interface.*

2.4. *Software Engineering Body of Knowledge* (SWEBOK)

SWEBOK adalah sebuah panduan yang dihasilkan dari sebuah proyek gagasan IEEE *Computer Society*. Panduan ini disusun sejak tahun 1998 dimana tim tersebut mulai menyusun pemahaman standart tentang bidang ilmu softwate engineering. Terdapat 5 tujuan utama pada SWEBOK, yaitu: (Society, 2014)

a. Untuk memperlihatkan kesamaan pandangan tentang rekayasa sistem di seluruh dunia.
b. Untuk memperjelas tempat dan menetapkan batas dari rekayasa sistem dan hubungan dengan disiplin ilmu lain seperti ilmu komputer, manajemen proyek, teknik komputer dan matematika.
c. Untuk membuat karakter isi dari disiplin ilmu rekayasa sistem.
d. Untuk memberikan akses topik ke SWEBOK.
e. Untuk memberikan pengetahuan dasar bagi pengembangan kurikulum dan sertifikasi serta perijinan.

Salah satu model dalam SDLC adalah model *prototype*. Menurut Pressman (2012) dalam pembuatan prototipe (*prototyping*) seringkali pelanggan mendefinisikan sejumlah sasaran perangkat lunak secara umum, tetapi tidak bisa mengidentifikasi spesialisasi kebutuhan yang rinci untuk fungsi-fungsi dan fitur-fitur yang nantinya akan dimiliki perangkat lunak yang akan dikembangkan. Dalam kasus lain, pengembang perangkat lunak mungkin merasa tidak pasti tentang efisiensi suatu algoritma yang akan digunakan dalam pengembangan perangkat lunak, atau juga sistem operasi yang akan digunakan, atau merasa tidak pasti akan bentuk interaksi banyak situasi yang lain, paradigma pembuatan prototipe (*prototyping*) mungkin menawarkan pendekatan yang paling baik.



Gambar 2.1 Model Prototipe

Langkah-langkah dalam pengerjaan model *prototype* adalah sebagai berikut:

2.4.1. *Communication*

Communication merupakan tahapan dalam pengumpulan data kebutuhan. Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasi semua kebutuhan, dan garis besar sistem yang akan dibuat. Dalam tahapan *communication* akan dilakukan dengan konsep SWEBOK yaitu *Software Requirement*.

Software requirement adalah kebutuhan perangkat lunak dapat diartikan sebagai properti yang harus ditampilkan dalam rangka memecahkan beberapa masalah di dunia nyata. Area pengetahuan dari *software requirement* adalah

elisitasi, analisis, spesifikasi, dan validasi persyaratan perangkat lunak. (Society, 2014)

Softwatre Requirement menghasilkan informasi tentang desain yang akan menjadi dasar, sehingga dapat mengetahui dimana sebuah sistem akan digunakan, oleh siapa, dan layanan apa yang harus disediakan. Berikut ini adalah tahapan dalam *software requirement*:

A. *Requirement Process*

Tahap proses kebutuhan merupakan tahapan yang memperkenalkan proses kebutuhan dari perangkat lunak yang menggambarkan bagaimana proses kebutuhan *dovetails* dengan proses rekayasa perangkat lunak secara keseluruhan. Pada tahap ini dilakukan proses model dan proses aktor yang terdiri dari beberapa aktivitas yaitu:

A.1. Analisis *Elimination-Simplification-Integration-Automation* (ESIA)

Analisis ESIA merupakan analisis yang digunakan untuk melakukan perubahan dalam arti perbaikan proses yang dapat dilakukan dalam berbagai bentuk yang secara garis besar dapat berupa: (Indrajit & Djokopranoto, 2002)

A.1.1. *Elimintaion*

Proses eliminasi merupakan proses yang digunakan untuk menghilangkan suatu proses yang tadinya ada menjadi tidak ada karena dianggap tidak perlu atau perlu diganti dengan proses yang sama sekali baru.

A.1.2. *Simplification*

Proses menyederhanakan merupakan proses yang digunakan untuk menyederhanakan dengan melakukan berbagai cara seperti *benchmarking* atau menggunakan jasa konsultan.

A.1.3. *Integration*

Proses integrasi yaitu menggabungkan beberapa proses menjadi satu proses. Pada hakikatnya sama dengan menyederhanakan, namun lebih spesifik sifatnya.

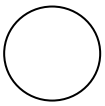
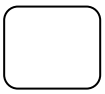
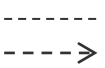
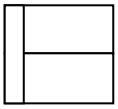
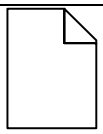
A.1.4. *Automation*

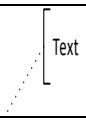
Proses automasi digunakan untuk meningkatkan kecepatan, ketelitian, dan efisiensi. Hal ini digunakan dengan menggunakan jasa komputer atau teknologi informasi.

A.2. *Business Process Modeling Notation* (BPMN)

Business Proess Management Initiative (BPMI) telah mengembangkan standar *Business Process Modeling Notation* (BPMN). Tujuan utama dari BPMN adalah untuk memberikan notasi yang mudah dipahami oleh semua pengguna bisnis, dari analis bisnis yang menciptakan konsep awal dari proses, untuk para pengembang teknis yang bertanggung jawab untuk melaksanakan proses-proses teknologi tersebut, dan akhirnya, kepada orang-orang bisnis yang akan mengelola dan memantau proses tersebut. Berikut adalah notasi-notasi dalam BPMN sebagai berikut: (Group, 2008)

Tabel 2.1 Notasi BPMN

Element	*Description*	*Notation*
Event	*Event* adalah sesuatu yang "terjadi" selama proses bisnis. Peristiwa ini mempengaruhi aliran proses dan biasanya memiliki penyebab (trigger) atau dampak (hasil). Acara yang lingkaran dengan pusat terbuka untuk memungkinkan penanda internal untuk membedakan pemicu atau hasil yang berbeda. Ada tiga jenis Events, berdasarkan kapan mereka mempengaruhi aliran: Mulai, Menengah, dan End.	
Activity	*Activity* adalah istilah umum untuk pekerjaan yang perusahaan lakukan. Suatu kegiatan dapat atom atau non-atom (compound). Jenis-jenis kegiatan yang merupakan bagian dari Model Proses adalah: Proses, Sub-Proses, dan Task. Tugas dan Proses Sub adalah persegi panjang bulat. Proses yang terkandung dalam kolam.	
Gateway	*Gateway* digunakan untuk mengontrol perbedaan dan konvergensi Urutan Arus. Dengan demikian, akan menentukan bercabang, forking, penggabungan, dan bergabung jalur. Penanda internal yang akan menunjukkan jenis kontrol perilaku.	
Sequence Flow	*Sequence Flow* digunakan untuk menunjukkan urutan kegiatan akan dilakukan di sebuah proses	
Message Flow	*Message Flow* digunakan untuk menunjukkan aliran pesan antara dua peserta yang siap untuk mengirim dan menerima mereka. Dalam BPMN, dua *pools* terpisah dalam diagram akan mewakili dua peserta (misalnya, badan usaha atau peran bisnis).	
Assosiation	*Assosiation* digunakan untuk menghubungkan informasi dengan arus *objects*. Teks dan grafis non-Arus objek dapat dikaitkan dengan arus *objects*. Sebuah panah dari Asosiasi menunjukkan arah aliran (misalnya, data), saat yang tepat.	
Pool	*Pool* merupakan peserta dalam proses juga bertindak sebagai "*swimlane*" dan wadah grafis untuk partisi serangkaian kegiatan dari *pools* lain, biasanya dalam konteks situasi B2B.	
Lane	*Lane* adalah sub-partisi dalam pool dan akan memperpanjang seluruh panjang pool, baik secara vertikal maupun horizontal. Lane yang digunakan untuk mengatur dan mengkategorikan kegiatan.	
Data Object	*Data Objects* dianggap artefak karena mereka tidak memiliki efek langsung pada *Sequence Flow* atau *Message Flow* proses, tapi mereka memberikan informasi tentang kegiatan apa perlu dilakukan dan / atau apa yang mereka hasilkan.	
Group	Sebuah pengelompokan kegiatan yang berada dalam	

Element	*Description*	*Notation*
	kategori yang sama. Jenis pengelompokan tidak mempengaruhi Sequence Flow kegiatan dalam kelompok. Nama kategori muncul pada diagram sebagai label kelompok. Kategori dapat digunakan untuk dokumentasi atau analisis tujuan. Kelompok adalah salah satu cara di mana kategori objek dapat secara visual ditampilkan pada diagram.	
Text Annotation	*Text Annotation* mekanisme untuk modeler untuk memberikan informasi tambahan bagi pembaca dari Diagram BPMN.	

B. *Requirement Elicitation*

Tahap elisitasi kebutuhan adalah tahap yang digunakan untuk melakukan komunikasi secara efektif antara berbagai pemangku kepentingan. Selanjutnya, dari komunikasi ini dilanjutkan ke proses *Software Development Life Cycle* (SDLC). Proses ini adalah proses yang sangat penting sebelum pembangunan perangkat lunak dimulai. *Element* penting lain dari persyaratan elisitasi adalah bagaimana ruang lingkup dari proyek yang akan dikerjakan. Dalam tahap ini akan dilakukan dengan menggunakan metode *interviews* dan *observations*.

C. *Requirement Analysis*

Requirement analysis adalah tahapan yang digunakan untuk mempelajari kebutuhan pengguna, sehingga didapatkan definisi kebutuhan sistem atau perangkat lunak yang bertujuan untuk mendefinisikan apa yang harus dikerjakan oleh perangkat lunak dalam memenuhi keinginan pengguna dan memahami masalah secara menyeluruh. Pada tahap ini secara detil terdapat proses sebagai berikut:

C.1. *Requirement Classification*

Dalam tahap klasifikasi kebutuhan, terdapat beberapa pengklasifikasian kebutuhan berdasarka pada sejumlah dimensi seperti analisis kebutuhan pengguna, analisis data, dan analisis metode.

C.1.1. Analisis Kebutuhan Pengguna

Dalam melakukan analisis kebutuhan pengguna akan digunakan konsep dari *stakeholder requirement*. *Stakeholder requirement* pernyataan dari kebutuhan *stakeholder* tertentu atau kelas *stakeholder* yang menggambarkan kebutuhan dari pemangku kepentingan dan bagaimana pemangku kepentingan akan berinteraksi dengan solusi.

C.1.2. Analisis Data

Dalam melakukan analisis data akan digunakan konsep analisis otokeralasi dengan bantuan dari *software* minitab.

a. Minitab

Paket program Minitab merupakan perangkat lunak statistika yang dapat digunakan sebagai media pengolahan data yang menyediakan berbagai jenis perintah yang memungkinkan proses pemasukan data, manipulasi data, pembuatan grafik, peringkasan nilai-nilai numerik, dan analisis statistika lainnya.

Minitab adalah program komputer yang dirancang untuk melakukan pengolahan statistik. Minitab mengkombinasikan kemudahan penggunaan layaknya Microsoft Excel dengan kemampuan melakukan analisis statistika yang

kompleks. Minitab 16, versi terbaru perangkat lunak ini, tersedia dalam tujuh bahasa: Inggris, Perancis, Jerman, Jepang, Korea, Mandarin, dan Spanyol. (Imanda, 2010)

b. Analisis Otokorelasi

Analisis otokeralasi digunakan untuk mengetahui pola data dari suatu data runtut waktu apakah data tersebut bersifat trend, musiman, dan ketidakberaturan. Koefisien-koefisien otokorelasi untuk setiap variabel lamban (*lagged variable*) yang berbeda digunakan untuk mengidentifikasi pola data runtut waktu. Berikut merupakan persamaan untuk menghitung koefisien otokorelasi tingkat pertama (r1) atau korelasi antara Y_t dengan Y_{t-1}.

$$e_1 = \frac{\sum_{t=1}^{n-1}(Y_{t-1} - \bar{Y})(Y_{t-1} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2} \quad \text{..... (1)}$$

dimana:

r_1 = Koefisien otokorelasi tingkat pertama

$\bar{Y}$ = Nilai rata-rata serial data

Y_t = Observasi pada waktu t

Y_{t-1} = Observasi pada satu periode sebelumnya

C.1.3. Analisis Metode Peramalan

a. Peramalan

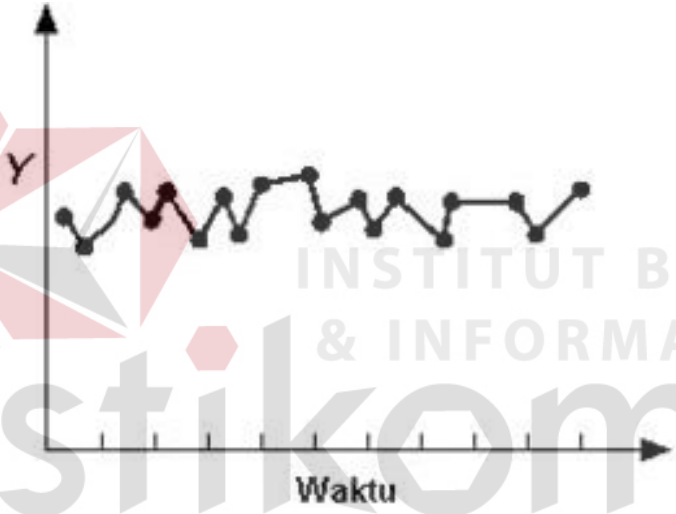
Peramalan adalah sebuah kegiatan sebelum perencanaan yang bertujuan memperkirakan kondisi pasar dan permintaan konsumen dimasa mendatang. Peramalan ini penting karena organisasi makin kompleks untuk pengambilan

keputusan keadaan lingkungan dan keinginan konsumen berbuah cepat. (Martono, 2013)

b. Data Runtut Waktu

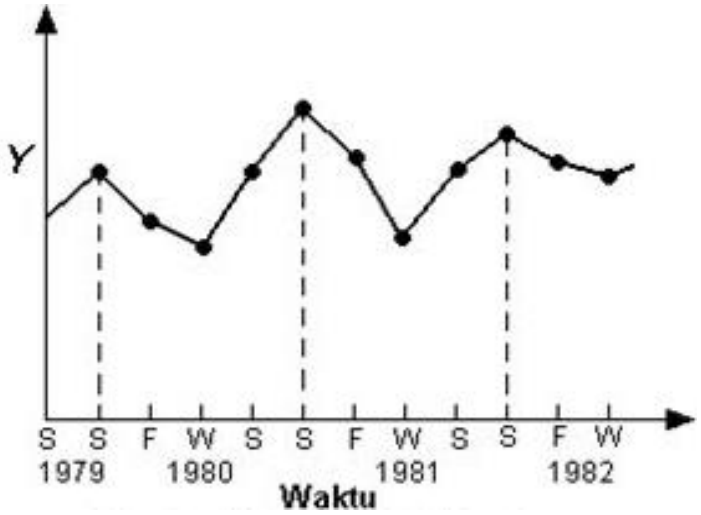
Menurut Makridakis, Wheelwright (2000), pola data time series dapat dibedakan menjadi empat yaitu:

a. Pola horizontal terjadi apabila nilai dari data mengalami fluktuasi didaerah nilai rata-rata konsumen (nilai rata-ratanya stasioner).



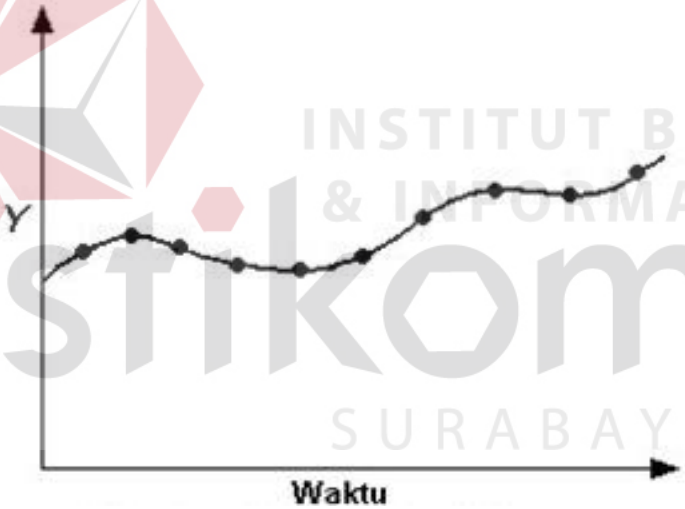
Gambar 2.2 Pola Stasioner

b. Pola musiman terjadi apabila suatu deret dari data dipengaruhi oleh faktor musiman yang ditunjukkan oleh adanya pola yang teratur yang bersifat musiman.



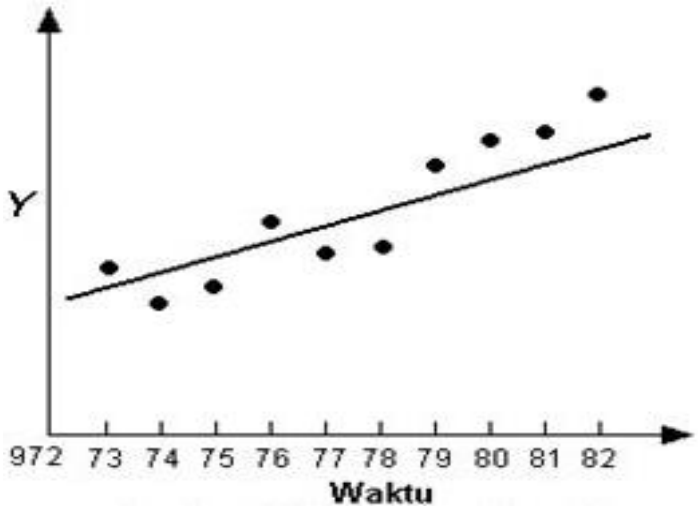
Gambar 2.3 Pola Musiman

c. Pola siklis terjadi apabila pola data deret waktu mengalami fluktuasi ekonomi jangka panjang berhubungan dengan siklis bisnis.



Gambar 2.4 Pola Siklis

d. Pola trend terjadi apabila pola data mengalami kenaikan atau penurunan, pola data seperti ini bervariasi tak beraturan.



Gambar 2.5 Pola Tren

c. Pengukuran Kesalahan Peramalan

Menurut Arysad (2010) terdapat beberapa metode telah digunakan untuk menunjukkan kesalahan yang disebabkan oleh suatu teknik peramalan tertentu. Hampir semua ukuran tersebut menggunakan pengrata-rataan beberapa fungsi dari perbedaan antara nilai sebenarnya dengan nilai peramalannya. Perbedaan antara nilai sebenarnya dengan nilai peramalannya disebut *residual*. Berikut adalah persamaan untuk menghitung kesalahan atau *residual* dari setiap periode peramalan:

$$e_t = Y_t - \hat{Y}_t$$

..... (2)

di mana:

e_t = kesalahan peramalan pada periode t

Y_t = nilai sebenarnya pada periode t

$\hat{Y}_t$ = nilai peramalan pada periode t

Terdapat beberapa cara untuk mengevaluasi teknik peramalan yaitu:

a. ***Mean Absolute Deviation*** **(MAD)** mengukur akurasi peramalan dengan merata-ratakan kesalahan peramalan (nilai absolutnya). Cara ini sangat berguna untuk mengukur kesalahan peramalan dalam unit ukuran yang sama seperti data aslinya.

$$MAD = \frac{\sum_{t-1}^{n}(Y_t - \hat{Y}_t)}{n}$$

..... (3)

b. ***Mean Squared Error*** **(MSE)** merupakan metode alternatif dalam mengevaluasi suatu teknik peramalan. setiap kesalahan atau residual dikuadratkan, kemudian dijumlahkan dan dibagi dengan jumlah observasi.

$$MSE = \frac{\sum_{t-1}^{n}(Y_t - \hat{Y}_t)^2}{n}$$

..... (4)

c. ***Mean Absolute Percentage Error*** **(MAPE)** dihitung dengan menemukan kesalahan absolut setiap periode, kemudian membaginya dengan nilai observasi pada periode tersebut, dan akhirnya merata-ratakan persentase absolute ini.

$$MAPE = \frac{\sum_{t-1}^{n}\frac{|Y_t - \hat{Y}_t|}{Y_t}}{n}$$

..... (5)

d. ***Mean Percentage Error*** **(MPE)** dihitung dengan cara menemukan kesalahan setiap periode, kemudian membaginya dengan nilai sebenarnya pada periode tersebut, dan kemudian merata-ratakan persentase kesalahan tersebut.

$$MPE = \frac{\sum_{t-1}^{n}\frac{(Y_t - \hat{Y}_t)}{Y_t}}{n}$$

..... (6)

d. *Single Exponential Smoothing*

Pemulusan eksponensial (*exponential smoothing*) adalah suatu *tipe* teknik peramalan rata-rata bergerak yang melakukan penimbangan terhadap data masa lalu dengan cara eksponensial sehingga data paling akhir mempunyai bobot atau timbangan lebih besar dalam rata-rata bergerak. (Prasetya dan Lukiastuti, 2009)

Menurut Prasetya dan Lukiastuti (2009) pemulusan eksponensial sederhana tidak memperhitungkan pengaruh trend, sehingga tidak ada nilai α yang akan sepenuhnya mengantikan nilai trend dalam data. Nilai-nilai α rendah akan menyebabkan jarak yang lebih lebar dengan trend, karena hal itu memberikan bobot yang lebih kecil pada permintaan sekarang. Persamaan metode pemulusan eksponensial sederhana (*single exponential smoothing*) yang digunakan adalah sebagai berikut:.

$$\hat{Y}_{t+1} = \alpha Y_t + (1-\alpha)\hat{Y}_t \quad \text{..... (7)}$$

di mana:

$\hat{Y}_{t+1}$ = nilai ramalan untuk periode berikutnya

A = Konstanta pemulusan

Y_t = Data baru atau analisis nilai Y sebenarnya pada periode t

$\hat{Y}_t$ = Nilai pemulusan yang lama atau rata-rata pemulusan hingga periode t-1

e. Kekuatan dan Keterbatasan Teknik *Exponential Smoothing*

Menurut Arsyad (2010) dalam penerapannya, suatu metode pasti memiliki kekuatan dan keterbatasan dalam melakukan proses peramalan. Beberapa kekuatan dan keterbatasan teknik pemulusan eksponensial yaitu:

Tabel 2.2 Kekuatan dan Keterbatan Teknik *Exponential Smoothing*

No	Kekuatan	Keterbatasan
1	Relatif sederhana dan biaya rendah.	Hasil ramalan dengan teknik pemulusan eksponensial sangat sensitif terhadap spesifikasi konstanta pemulusan.
2	Hanya membutuhkan data terakhir yang dimuluskan, pasti akan menghemat biaya yang tidak sedikit.	Menghilangkan lag dibelakang titik balik dari suatu data runtut waktu yang aktual.
3		Model-model eksponensial sangat berguna jika tujuannya untuk peramalan jangka menengah dan pendek.
4		Ramalan-ramalan dapat mengandung kesalahan karena fluktuasi random yang sangat besar pada periode-periode terakhir yang mendapat bobot yang lebih berat.

C.1.4. Analisis Metode Perencanaan Persediaan

a. Pengertian *Stock Opname*

Stock opname merupakan kegiatan pengecekan terhadap obat atau perbekalan farmasi yang bertujuan untuk mengetahui jumlah dan jenis obat yang paling banyak digunakan untuk kebutuhan pemesanan dan untuk mencocokan antara jumlah obat yang ada di gudang dengan yang ada pada catatan. (Febriawati, 2013)

b. Biaya Penyimpanan

Biaya penyimpan (*holding cost*) adalah biaya yang berhubungan dengan penyimpanan atau "membawa" persediaan dari waktu ke waktu. Biaya penyimpanan juga meliputi biaya barang yang menjadi usang dan biaya yang berkaitan dengan gudang, seperti asuransi, karyawan tambahan, dan pembayaran bunga. (Tanuwijaya & Setyawan, 2012)

c. Kapasitas Gudang

Salah satu yang sangat mempengaruhi berfungsi atau tidaknya sebuah gudang atau kapasitas gudang tersebut. Dalam menentukan kapasitas gudang, maka keadaan yang harus dipertimbangkan adalah keadaan maksimum. Gudang mencapai keadaan maksimum pada saat sediaan pengemas belum dipakai, terjadi keterlambatan pemakaian bahan atau produk. (Lachman & Lieberman, 2008)

d. *Service Level*

Service Level merupakan ukuran mengenai seberapa baik bagian tersebut mampu mengisi kembali tingkat inventori barang atau tingkat pemenuhan kebutuhan inventori pada bagian lain yang membutuhkan. Ada periode dimana inventori barang jadi tidak mencukupi untuk memenuhi kebutuhan penjualan. Akibatnya, diperlukan waktu tambahan bagi konsumen untuk menunggu pengiriman barang jadi pada periode berikutnya.

Salah satu definisi *service level* adalah *Service Level Tipe* 1 (SL-1) yang menentukan *safety stock inventory* untuk mencapai *service level* yang dikehendaki. Untuk mengantisipasinya kemungkinan kehabisan inventori, jumlah permintaan pada tingkat probabilitas termasuk permintaan selama periode pengisian kembali inventori. Perhitungan ini menggunakan variabel penyesuaian (*safety factor*[1]) sebagai berikut:

Tabel 2.3 *Service Level*

Service Level	*Safety Factor*
50,00	0,00
75,00	0,67
80,00	0,84
84,13	1,00
85,00	1,04
89,94	1,25

Service Level	*Safety Factor*
97,72	2,00
98,00	2,05
98,61	2,20
99,00	2,33
99,18	2,40
99,38	2,50

Service Level	*Safety Factor*
90,00	1,28
91,00	1,34
93,32	1,50
94,52	1,60
95,00	1,65
96,00	1,75
97,00	1,88

Service Level	*Safety Factor*
99,60	2,65
99,70	2,75
99,80	2,88
99,86	3,00
99,90	3,09
99,93	3,20
99,99	4,00

Safety stock yang harus tersedia = *safety factor* x *standart deviation* dari kebutuhan *inventory*.

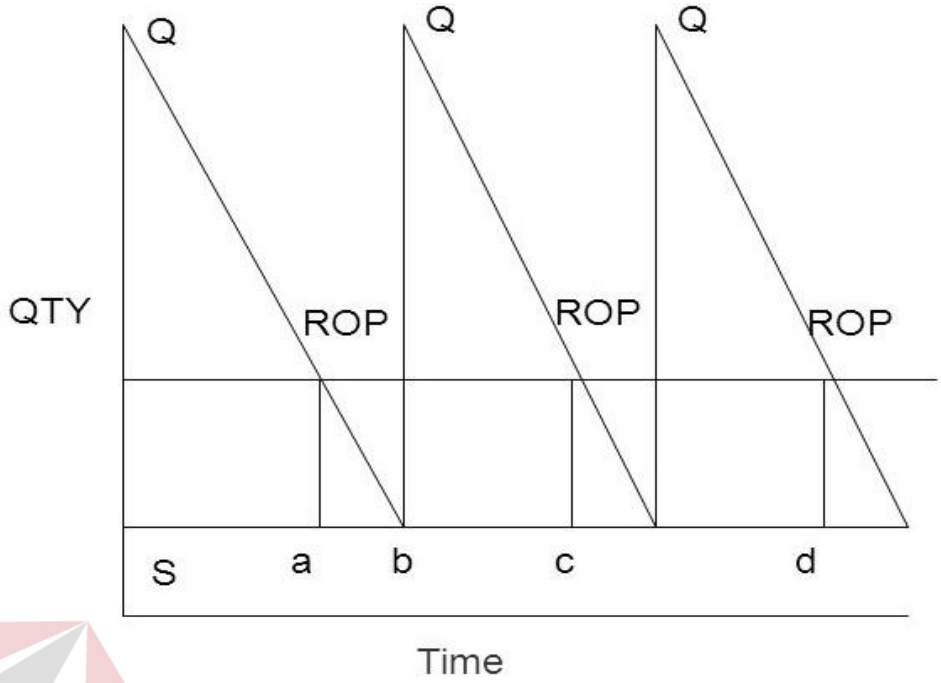
Perusahaan bisa menentukan sendiri service level yang diinginkan, bergantung kebijakan perusahaan atau mengikuti standart industri.

$$Standart\ Deviation = \sqrt{\frac{\sum(Kebutuhan - rata - rata\ kebutuhan)^2}{Jumlah\ periode - 1}} \quad \text{.....(8)}$$

Periode bisa dihitung harian, mingguan, atau bulanan, bergantung kebutuhan perusahaan. Semakin detail akan semakin baik (perhitungan periode harian lebih baik daripada mingguan). (Martono, 2013)

e. *Reorder Point* (ROP)

Reorder Point (ROP) adalah titik waktu dimana pemesanan dilakukan kembali, setelah persediaan mencapai jumlah tertentu, sehingga tidak terjadi kekurangan barang (bahan baku, komponen).



Gambar 2.6 Kurva Titik Pemesanan Kembali

Faktor utama yang perlu diperhatikan dalam menentukan ROP adalah kebutuhan rata-rata periode dan *lead time*, yaitu waktu antara dilakukannya pemesanan barang sampai dengan barang tersebut datang. (Tanuwijaya & Setyawan, 2012)

$$ROP = (D \; x \; L) + SS$$ .....(9)

dimana:

D = Permintaan kebutuhan barang per satuan waktu

L = Lead Time

SS = Safety Stock

f. *Economic Order Quantity* (EOQ)

Economic Order Quantity (EOQ) atau kuantitas pesanan order adalah salah satu teknik kontrol persediaan yang tertua dan paling dikenal, teknik ini relatif mudah digunakan, tetapi berdasarkan pada beberapa asumsi. (Heizer & Render, 2010)

Dengan model EOQ, kuantitas pesanan optimal akan muncul pada suatu titik dimana biaya penyetelan totalnya sama dengan biaya penyimpanan total. Berikut adalah langkah-langkah yang diperlukan dalam metode EOQ:

a. Mengembangkan seluruh pernyataan untuk biaya penyetelan atau pemesanan.

$$\text{Biaya penyetelan tahun} = \begin{pmatrix}\text{Jumlah pesanan}\\ \text{per tahun}\end{pmatrix} \text{x} \begin{pmatrix}\text{Biaya penyetelan atau}\\ \text{pesanan per pesanan}\end{pmatrix}$$

$$= \left(\frac{\text{Permintaan tahunan}}{\begin{matrix}\text{Jumlah unit dalam}\\ \text{setiap pesanan}\end{matrix}}\right) \begin{pmatrix}\text{Biaya penyetelan atau}\\ \text{pesanan per pesanan}\end{pmatrix}$$

$$= \left(\frac{D}{Q}\right)(S) = \frac{D}{Q}S$$

.....(10)

b. Mengembangkan sebuah pernyataan untuk biaya penyimpanan.

$$= (\text{Tingkat persediaan rata} - \text{rata}) \text{x} (\text{biaya penyimpanan per unit per tahun})$$

$$= \frac{\text{Kuantitas pesanan}}{2} (\text{Biaya penyimpanan per unit per tahun})$$

$$= \left(\frac{Q}{2}\right)(H) = \frac{Q}{2}H$$

.....(11)

c. Menentukan biaya penyetelan sama dengan biaya penyimpanan.

$$\frac{D}{Q}S = \frac{Q}{2}H$$

.....(12)

d. Selesaikan persamaan untuk kuantitas pesanan optimal.

$$2DS = Q^2H$$

$$Q^2 = \frac{2DS}{H}$$

$$Q^* = \sqrt{\frac{2DS}{H}}$$

.....(13)

dimana:

Q = Jumlah unit per pesanan

Q^* = Jumlah optimum unit per pesanan (EOQ)

D = Permintaan tahunan dalam unit untuk barang persediaan

S = Biaya penyetelan atau pesanan untuk setiap pesanan

H = Biaya penyimpanan atau penyimpanan per unit per tahun

g. *Min-Max*

Min-Max inventory merupakan metode yang berlainan dengan konsep EOQ dan formula perencanaan berkala, konsep persediaan dan maksimum tidak berdasarkan pada perhitungan berskala, tetapi dapat dilakukan setiap waktu, dengan konsep titik pesan kembali atau *reorder point* untuk persediaan minimum. Sedangkan persediaan maksimum adalah sebanyak yang secara ekonomis mencapai optimal, yaitu sesuai dengan perhitungan EOQ. (Indrajit dan Djokopranoto, 2011)

$$Q = Max - Min$$

.....(14)

dimana:

Q = Jumlah yang dipesan untuk pengisian persediaan kembali

Max = *Maximum stock*

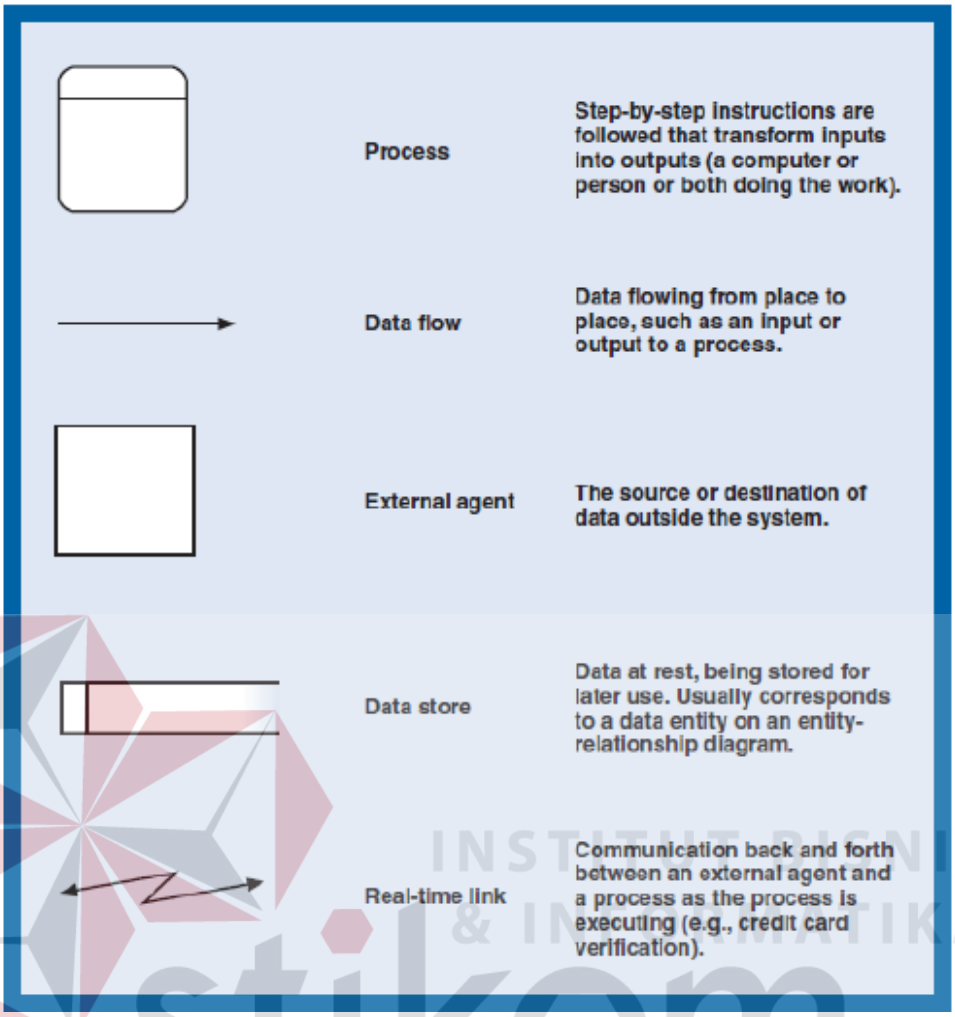
Min = *Minimum stock*

C.2.*Conceptual Modeling*

Pengembangan model dari masalah dunia nyata adalah kunci untuk analisis kebutuhan perangkat lunak. Tujuan mereka adalah untuk membantu dalam memahami situasi di mana masalah terjadi, serta menggambarkan solusi. Oleh karena itu, model konseptual terdiri model entitas dari domain masalah, dikonfigurasi untuk mencerminkan hubungan dunia nyata dan dependensi. Beberapa jenis model dapat dikembangkan antara lain yaitu *Data Flow Diagram* (DFD), *Entity Relationship Diagram* (ERD), dan *Conceptual Data Model* (CDM).

C.2.1. *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) merupakan model proses yang digunakan untuk menggambarkan aliran data *input* dan *output* pada sebuah sistem beserta tugas atau pengolahan yang dilakukan oleh sistem dari segi proses dan data *store*. Simbol-simbol yang dipakai oleh *data flow diagram* terdiri dari beberapa notasi seperti pada Gambar 2.7.



Gambar 2.7 *Data Flow Diagram Symbols*

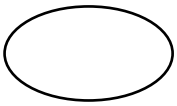
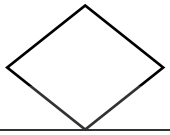
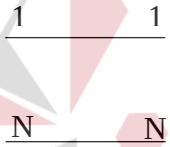
Data flow diagram dapat dikembangkan dalam kerincian yang bervariasi, sesuai *system requirement* yang perlu digambarkan. *Level* abstraksi DFD dibagi menjadi dua yaitu diagram konteks dan *fragmen* DFD. (Satzinger, Jakson, & Burd, 2010)

C.2.2. *Entity Relationship Diagram* (ERD)

Model *Entity-Relathionship* yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi demgam atribut-atribut yang merepresentasikan seluruh fakta dari ‘dunia nyata’ yang kita tinjau,

dapat digambarkan dengan lebih sistematis dengan menggunakan Diagram E-R. Berikut merupakan notasi-notasi dalam Diagram E-R: (Fathansyah, 2012)

Tabel 2.4 Notasi Diagram E-R

No	Nama	Notasi	Deskripsi
1	Persegi Panjang		Menyatakan himpunan entitas
2	Eclipse		Menyatakan atribut (atribut yang berfungsi sebagai key yang digaris bawahi)
3	Belah Ketupat		Menyatakan himpunan relasi
4	Garis		Sebagai penghubung antara himpunan relasi dengan himpunan entitas dan himpunan entitas dengan atributnya
5	Kardinalitas		Kardinalitas relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka (1 dan 1 untuk relasi satu ke satu, dan N untuk relasi satu ke banyak, atau N dan N untuk relasi banyak ke banyak.

D. Requirement Specification

Requirement specificatin adalah sebuah kegiatan yang mengacu pada pembuatan dokumen yang dapat ditinjau secara sistematis, dievaluasi, dan disetujui. Pada tahap ini, sama sekali tidak dibahas bagaimana metode pengembangan yang akan dilakukan.

Dalam melakukan analisis kebutuhan spesifikasi dari sistem yang digunakan pada pengembangan sistem ini, yaitu analisis kebutuhan fungsional dan non fungsional sebagai berikut:

D.1.*Software Requirement Spesifications*

D.1.1. *Functional Requirement*

Functional requirements menggambarkan fungsi dari perangkat lunak untuk menjalankan; misalnya, format teks atau modulasi sinyal. Mereka kadang-kadang dikenal sebagai kemampuan atau fitur. Persyaratan fungsional juga dapat digambarkan sebagai salah satu yang himpunan terhingga langkah uji dapat ditulis untuk memvalidasi perilakunya.

D.1.2. *Non-functional Requirement*

Non-functional requirements (*NFR*) adalah orang-orang yang bertindak untuk membatasi solusi. persyaratan non fungsional kadang-kadang dikenal sebagai kendala atau persyaratan mutu. Beberapa kategori non fungsional yaitu sebagai berikut: 1) *Security*; 2) *Correctness*; 3) *Interface*; 4) *Performance*; dan 5) *Operability*

E. *Requirement Validation*

Diperlukan validasi dan verifikasi terhadap dokumen-dokumen persyaratan yang telah dibuat. Persyaratan-persyaratan divalidasi untuk menjamin bahwa enginerr perangkat lunak telah memahami persyaratan, serta perlu juga untk memverifikasi bahwa dokumen persyaratan telah sesuai dengan standar perusahaan dan dapat dimengeri, konsisten, serta lengkap. Proses validasi dan verifikasi ini melibatkan pengguna sebagai pihak yang menilai dan memberi *feedback* (umpan balik).

Dalam melakukan *requirement validation* terhadap sistem pada nantinya akan digunakan metode *prototyping*. Prototyping umumnya sarana untuk memvalidasi interpretasi *software engineering* dari kebutuhan perangkat lunak, serta untuk memunculkan kebutuhan baru. Seperti elisitasi, ada berbagai teknik prototyping dan sejumlah titik dalam proses dimana prototipe validasi mungkin tepat. Keuntungan dari prototipe adalah bahwa mereka dapat membuat lebih mudah untuk menafsirkan asumsi *software engineering* dan, di mana diperlukan, memberikan umpan balik yang berguna tentang mengapa mereka salah.

2.4.2. *Quick Plan and Modeling Quick Design*

Quick plan and modeling quick design merupakan tahapan untuk membangun *prototype* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat *input* dan format *output*). Dalam melakukan tahapan *quick plan and modeling quick design* akan digunakan konsep *Software Design*.

Software design adalah tahap yang memainkan peran penting dalam mengembangkan perangkat lunak. *Software design* adalah proses yang mendefinisikan arsitektur, komponen, interface, dan karakteristik sebuah sistem atau komponen lainnya serta hasil dari proses tersebut, sehingga dapat dikatakan bahwa *software design* adalah cetak biru dari solusi yang akan diimplementasikan. (Society, 2014)

Software design secara spesifik memiliki hubungan dengan *software requirement, software construction, software engineering management, software engineering model and methods, software quality*, dan *computing foundations*.

Disamping itu, *software design* berupaya menganalisis *input* data secara sistematis, memproses atau mentrasformasikan data, menyimpan data, dan menghasilkan *output* informasi. Berikut ini adalah tahapan dalam *software design*:

A. *Software Structure and Architecture*

Software structure and architecture adalah proses mendeskripsikan dan mendefinisikan bagaimana *software* dibentuk dan diorganisasikan ke dalam komponen-komponen yang akan membentuk *software* tersebut. Hasil dari proses ini adalah sekumpulan model yang mendeskripsikan tujuan serta gambaran dari *software* yang akan dibangun, adapun model-model ini bisa dibangun dengan menggunakan *modeling language* (bahasa yang digunakan untuk membuat dan menyajikan informasi atau *knowledge*).

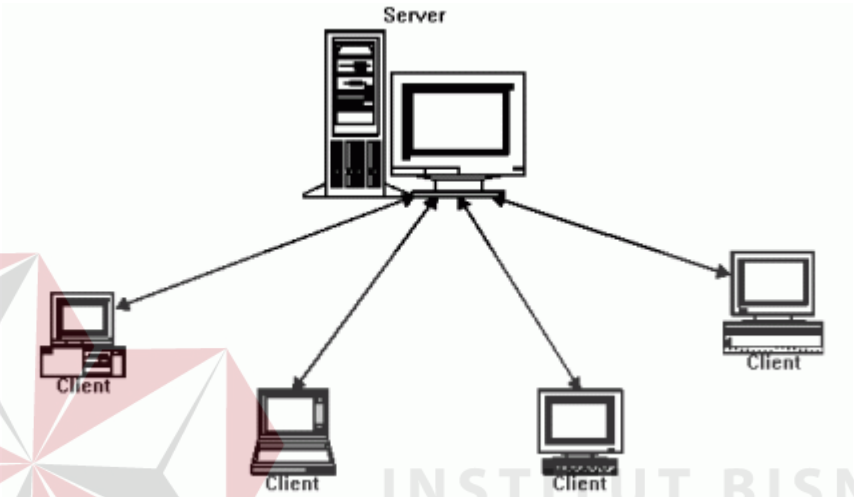
A.1.*Architectural Styles*

Gaya arsitektur adalah spesialisasi dari elemen dan hubungan jenis, bersama-sama dengan satu set kendala pada bagaimana mereka dapat digunakan. Sebuah gaya arsitektur dapat dipandang sebagai menyediakan tingkat tinggi organisasi perangkat lunak. Berbagai penulis telah mengidentifikasi sejumlah gaya arsitektur utama yaitu *client-server* dan *Model-View-Controller* (MVC).

A.1.1. *Client-Server*

Sesuai dengan namanya, jaringan komputer jenis ini memerlikan sebuah (atau lebih) komputer yang difungsikan sebagai pusat pelayanan dalam jaringan yang disebut server. Komputer-komputer lain disebut sebagai Client atau

Workstation. Sesuai sebutannya, komputer server bertugas melayani semua kebutuhan komputer lain yang berada dalam jaringan. Semua fungsi jaringan dikendalikan dan diatur oleh komputer server, termasuk masalah keamanan jaringan seperti hak akses data, waktu akses, sumber daya dan sebagainya. (Dulay, 2007)



Gambar 2.8 *Client-Server*

Keunggulan Jaringan *Client-Server:*

a. Memberikan keamanan yang lebih baik.
b. Lebih mudah pengaturannya bila *network* besar karena administrasinya disentralkan.
c. Semua data dapat di *backup* pada satu lokasi sentral.

Kelemahan Jaringan *Client-Server*:

a. Membutuhkan hardware yang lebih tinggi dan mahal untuk mesin *server*.
b. Mempunyai satu titik lemah jika menggunakan satu *server*, data *user* menjadi tak ada jika *server* mati.

A.1.2. *Model-View-Controller* (MVC)

Model-View-Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu Smalltalk (Trygve Reenskaug) untuk meng-enkapsulasi data bersama dengan pemrosesan (*model*), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk direpresentasikan pada sebuah user interface. Definisi teknis dari arsitektur MVC dibagi menjadi tiga lapisan yaitu: (Deacon, 2009)

a. *Model*

Model digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi yang berhubungan dengan pemrosesan data. Sebuah *model* meringkas lebih dari sekedar data dan fungsi yang beroperasi di dalamnya. Pendekatan *model* yang digunakan untuk komputer *model* atau abstraksi dari beberapa proses dunia nyata. Hal ini tidak hanya menangkap keadaan proses atau sistem, tetapi bagaimana sistem bekerja. Sebagai contoh, programmer dapat menentukan model yang menjembatani komputasi *back-end* dengan *front-end* GUI (*Graphical User Interface*).

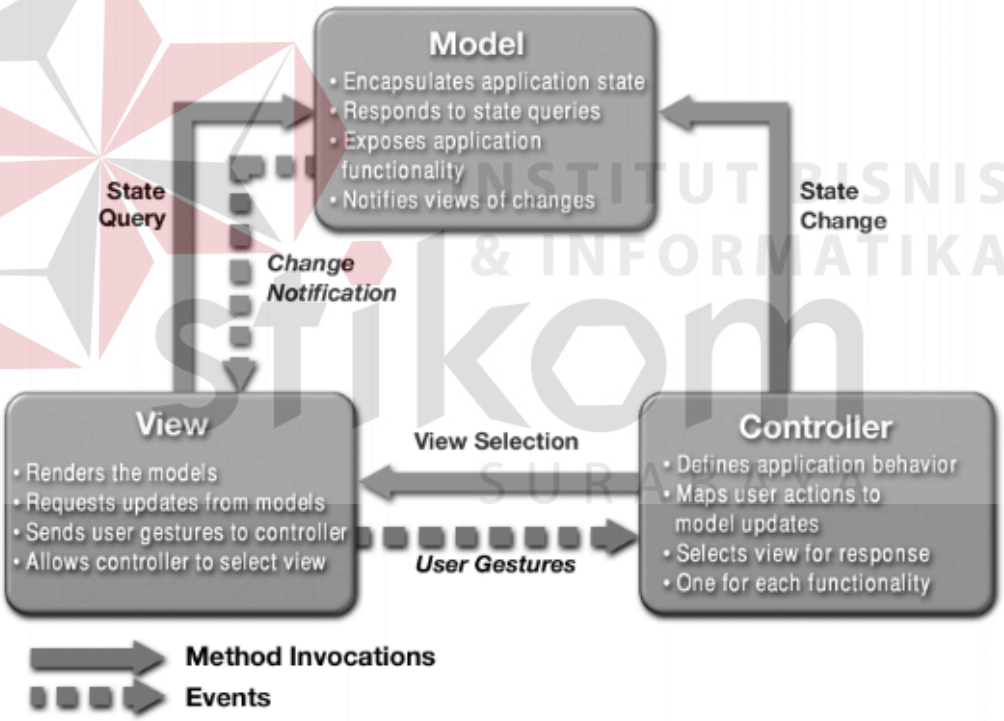
b. *View*

View bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. *View* biasanya memiliki hubungan 1-1 dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. *View* melekat pada model dan me-render isinya ke permukaan layar. Selain itu, ketika model berubah, view secara otomatis menggambar ulang bagian layar yang terkena perubahan untuk menunjukkan perubahan tersebut. Terdapat kemungkinan beberapa *view* pada *model* yang sama

dan masing-masing *view* tersebut dapat me-*render* isi *model* untuk permukaan tampilan yang berbeda.

c. *Controller*

Controller menerima *input* dari pengguna dan menginstruksikan model dan view untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, controller bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap respon aplikasi. Sebagai contoh, ketika pengguna mengklik tombol atau memilih item *menu*, *controller* bertanggung jawab untuk menentukan bagaimana aplikasi seharusnya merespon.



Gambar 2.9 Hubungan Antara *Model, View*, dan *Controller*

A.2.*Design Pattern*

Pola adalah solusi umum untuk masalah umum dalam konteks tertentu. Sementara gaya arsitektur dapat dilihat sebagai pola yang menggambarkan

organisasi tingkat tinggi dari perangkat lunak, pola desain lainnya dapat digunakan untuk menjelaskan rincian pada tingkat yang lebih rendah. pola desain tingkat yang lebih rendah ini meliputi PDM dan *Table Structured*.

B. *User Interface Design*

User interface design adalah bagian penting dari proses desain perangkat lunak. Desain user interface yang harus memastikan bahwa interaksi antara manusia dan mesin menyediakan untuk operasi yang efektif dan kontrol mesin. Untuk perangkat lunak untuk mencapai potensi penuh, antarmuka pengguna harus dirancang agar sesuai dengan kemampuan, pengalaman, dan harapan penggunanya diantisipasi. Beberapa teknik yang digunakan dalam melakukan *user interface design* sebagai berikut:

B.1. User Interface Design Process

Desain *user interface* merupakan proses berulang, prototipe *interface* sering digunakan untuk menentukan fitur, organisasi, dan melihat dari *interface* pengguna perangkat lunak. Proses ini mencakup tiga kegiatan utama yaitu 1) *User Analaysis*; 2) *Software Prototype*; dan 3) *Interface Evaluation*. Dalam melakukan ketiga proses tersebut terdapat beberapa prinsip yang harus dilakukan dalam menjalankannya yaitu:

B.1.1. *General UI Design Principles*

General UI design principles merupakan prinsip-prinsip desain UI mengenai *learnability*, *user familitary*, *consistency*, *minimal surprise*, *recoverability*, *user guidance*, dan *user diversity*.

B.1.2. *The Design of User Interaction Modalities*

Prinsip ini menjelaskan mengenai gaya interaksi antara *user* dengan sistem. Gaya interaksi ini dapat diklasifikasikan ke dalam gaya utama sebagai yaitu 1) *Question-Answer*; 2) *Direct Manipulation*; 3) *Menu Selection*; 4) *Form Fill-in*; 5) *Command Language*; dan 6) *Natural Language*.

B.1.3. *The Design of User Information Presentation*

Penyajian informasi mungkin tekstual atau grafis di alam. Sebuah desain yang baik membuat presentasi informasi yang terpisah dari informasi itu sendiri. Menurut gaya penyajian informasi, desainer juga dapat menggunakan warna untuk meningkatkan antarmuka. Ada beberapa pedoman penting yaitu 1) Batasi jumlah warna yang digunakan; 2) Gunakan perubahan warna untuk menunjukkan perubahan status perangkat lunak; dan 3) Gunakan warna untuk memudahkan akses bagi orang dengan buta warna atau kekurangan warna (misalnya, menggunakan perubahan saturasi warna dan kecerahan warna, cobalah untuk menghindari kombinasi biru dan merah).

C. *Software Design Notation*

Banyak notasi yang ada untuk mewakili artefak desain perangkat lunak. Beberapa digunakan untuk menggambarkan struktur organisasi dari desain, yang lain untuk mewakili perilaku perangkat lunak. notasi tertentu digunakan terutama selama desain arsitektur dan lain-lain terutama selama desain rinci, meskipun beberapa notasi dapat digunakan untuk kedua tujuan.

C.1. *Behavioral Description* (*Dynamic View*)

Berikut notasi dan bahasa, beberapa grafis dan beberapa tekstual, yang digunakan untuk menggambarkan perilaku dinamis dari sistem perangkat lunak dan komponen. Banyak dari notasi ini berguna terutama, tetapi tidak eksklusif, selama desain rinci. Selain itu, deskripsi perilaku dapat mencakup pemikiran untuk keputusan desain seperti bagaimana desain akan memenuhi kebutuhan keamanan. Dalam menggambarka *behavioral decription* akan digunakan teknik pseudocode, dimana pseudocode sendiri digunakan untuk menggambarkan secara umum pada tahap desain rinci, perilaku prosedure, atau metode.

D. *Software Design Strategies and Methods*

Berbeda dengan strategi umum, metode yang lebih spesifik dalam bahwa mereka umumnya menyediakan satu set notasi untuk digunakan dengan metode, deskripsi proses yang akan digunakan ketika mengikuti metode, dan seperangkat pedoman untuk menggunakan metode ini. metode tersebut berguna sebagai kerangka umum untuk tim *software engineering*. Salah satu *strategies design and method* yang digunakan dalam tahap ini adalah *Function-Oriented* (*Structured*)

Design. *Function-Oriented* merupakan salah satu metode klasik software desain, di mana pusat dekomposisi pada identifikasi fungsi perangkat lunak utama dan kemudian mengelaborasi dan menyempurnakan mereka dengan cara topdown hirarkis. Dalam penggambarannya akan digunakan *Structured Charts*.

2.4.3. *Construction Prototipe*

Pada tahap ini dijelaskan mengenai pembuatan prototipe berdasarkan hasil dari tahap *quick plan and modeling quick design*. Dalam pembuatan prototipe akan menggunakan konsep dari SWEBOK yaitu *Software Construction*.

Pada *software constrruction* ialah melakukan konversi hasil desain ke aplikasi yang lengkap melalui tahapan coding atau pengkodean termasuk bagaimana membuat basis data dan menyiapkan prosedur kasus pengujian, mempersiapkan berkas atau file pengujian, pengkodean pengompilasian, memperbaiki dan membersihkan program serta melakukan peminjaman pengujian. Construction ini memiliki beberapa tahapan secara umum. (Society, 2014)

E. *Practical Considerations*

Konstruksi adalah kegiatan di mana insinyur perangkat lunak harus berurusan dengan kendala dunia nyata, dan dia harus melakukannya dengan tepat, karena akan berpengaruh kepada konstruksi lebih didorong oleh pertimbangan praktis dari beberapa kas lain, dan rekayasa perangkat lunak yang mungkin paling *craftlike* dalam kegiatan konstruksi.

A.3.*Construction Design*

Beberapa proyek mengalokasikan kegiatan desain yang cukup besar untuk pembangunan, sementara yang lain mengalokasikan desain untuk fase secara eksplisit berfokus pada desain. Terlepas dari alokasi yang tepat, beberapa pekerjaan desain rinci akan terjadi pada tingkat konstruksi, dan bahwa karya desain cenderung didikte oleh kendala yang diberlakukan oleh masalah dunia nyata yang sedang ditangani oleh perangkat lunak. Dalam melakukan tahap ini akan menggunakan beberapa tools yaitu 1) *SqlServer*; 2) VB.Net; 3) *Coding*; 4); *Exception, Error Handling,* dan *Fault Tolerance*; dan 5) GUI *Builder*.

2.4.4.*Deployment, Delivery and Feedback*

Pada tahap *deployment delivery & feedback* dilakukan evaluasi oleh pelanggan apakah prototipe yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Evaluasi dilakukan dengan melakukan pengujian sistem dan hasil evaluasi pelanggan untuk mendapatkan *feedback*. Apabila hasil prototype tidak disetujui, maka proses akan kembali pada tahap satu (*Communication*). *Deployment delivery and feedback* akan dilakukan menggunakan konsep *Software Testing* dari SWEBOK.

Software Testing meliputi verifikasi yang dinamis dari tingkah laku sebuah sistem yang diwakili oleh beberapa contoh kasus uji coba. Kasus uji coba tersebut dilakukan dengan memberikan masukan kepada sistem agar muncul tingkah laku/reaksi yang diharapkan, begitu pula sebaliknya. Dalam uji coba sistem, terdapat hal-hal yang harus diperhatikan, yaitu: (Society, 2014)

F. *Test Level*

Test level dari uji coba. Didalamnya dijelaskan tentang target dari uji coba dan tujuan dari uji coba tersebut. Target dari tes dapat bervariasi: satu modul, sekelompok modul tersebut (terkait dengan tujuan, penggunaan, perilaku, atau struktur), atau seluruh sistem. Tiga tahap pengujian dapat dibedakan: Unit, integrasi, dan sistem. Ketiga tahapan tes tidak menyiratkan model proses, juga salah satu dari mereka diasumsikan lebih penting dibandingkan dengan dua lainnya. Metode ini dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit testing juga. (Hill, 2009)