BAB II

LANDASAN TEORI

2.1 PT. Baba Rafi Indonesia

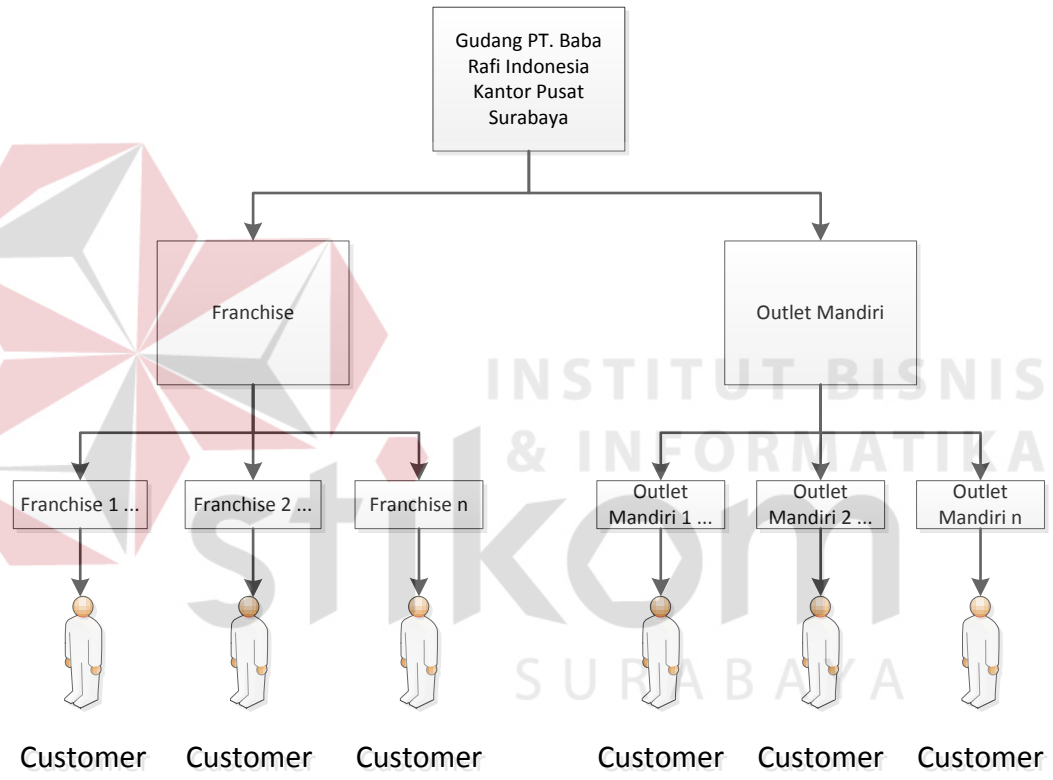
PT. Baba Rafi Indonesia merupakan sebuah waralaba (*franchise*) makanan cepat saji khas Timur Tengah. PT. Baba Rafi Indonesia memiliki 375 *outlet* yang tersebar di seluruh Nusantara. Pada saat ini PT. Baba Rafi Indonesia tidak hanya melebarkan sayap di Indonesia. Kini, PT. Baba Rafi Indonesia telah resmi terdaftar sebagai anggota *Malaysian Franchise Association* (MFA) dan siap melebarkan sayapnya ke Thailand, Vietnam, Brunei Darussalam serta Filipina.

PT. Baba Rafi memiliki visi dan misi. Visi PT. Baba Rafi Indonesia adalah menjadi perusahaan waralaba makanan cepat saji khas Timur Tengah terdepan di Indonesia dan Internasional. Misi PT. Baba Rafi Indonesia antara lain, yaitu:

1. Meningkatkan dan mengembangkan mutu dan pelayanan produk makanan cepat saji khas Timur Tengah.
2. Meningkatkan jumlah *franchise*.
3. Meningkatkan kualitas kontrol (*control quality*) *franchise*, pelayanan, dan produk.
4. Meningkatkan Budaya Kerja melalui peningkatan kualitas Sumber Daya Manusia (SDM) dan menerapkan *Good Corporate Governance* (GCG).
5. Meningkatkan *corporate value* dan *corporate image*.

Gambar 2.1 adalah alur distribusi pemasokan bahan baku Gudang PT. Baba Rafi Indonesia Kantor Pusat Surabaya. Gudang PT. Baba Rafi Indonesia

Kantor Pusat Surabaya memasok bahan baku ke semua waralaba dan *outlet* mandiri yang berada di Jawa Timur, Jawa Tengah, Bali, Cilacap, Kalimantan, dan Sulawesi. Jumlah *outlet* yang tersebar pada wilayah pemasokan bahan baku Gudang PT. Baba Rafi Indonesia Kantor Pusat Surabaya adalah 75 *outlet* waralaba dan 50 *outlet* mandiri. Saat ini sebagian besar *outlet* mandiri tidak aktif sehingga *outlet* mandiri yang aktif tinggal 20 *outlet*.



Gambar 2.1 Alur Distribusi Pemasokan Bahan Baku Gudang PT. Baba Rafi Indonesia Kantor Pusat Surabaya

Adapun penelitian ini membahas peramalan permintaan bahan baku *outlet* mandiri yang didistribusikan pada daerah Surabaya. Jumlah *outlet* mandiri yang tersebar pada daerah Surabaya adalah dua puluh *outlet*. Dua puluh *outlet* tersebut

tersebar di daerah Surabaya Selatan, Surabaya Timur, Surabaya Utara, Surabaya Barat, dan Surabaya Pusat.

2.2 Sistem

Sistem dapat didefinisikan dengan pendekatan prosedur dan dengan pendekatan komponen. Dengan pendekatan prosedur, sistem dapat didefinisikan sebagai kumpulan dari prosedur-prosedur yang mempunyai tujuan tertentu. Dengan pendekatan komponen, sistem dapat didefinisikan sebagai kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu (Jogiyanto, 2003).

Suatu sistem sebenarnya terdiri dari dua bagian, yaitu struktur dan proses. Struktur adalah komponen dari sistem tersebut dan proses adalah prosedurnya. Kedua pendekatan tersebut hanya mengambil satu aspek dari sistem saja untuk menjelaskannnya dari sudut pandangan aspek tersebut (Jogiyanto, 2003).

2.3 Sistem Perangkat Lunak

Perangkat keras komputer tidak akan dapat berbuat apa-apa tanpa adanya perangkat lunak. Teknologi yang canggih dari perangkat keras akan berfungsi bila instruksi-instruksi tertentu telah diberikan kepadanya. Instruksi-instruksi tersebut disebut dengan perangkat lunak (*software*). Perangkat lunak dapat diklasifikasikan ke dalam dua bagian besar (Jogiyanto, 2003), yaitu sebagai berikut ini.

1. Perangkat lunak sistem (*system software*), yaitu perangkat lunak yang dapat mengoperasikan sistem komputernya. Perangkat lunak sistem dapat dikelompokkan lagi menjadi empat bagian sebagai berikut ini.

a. Perangkat lunak sistem operasi (*operating system*), yaitu program yang ditulis untuk mengendalikan dan mengoordinasi operasi dari sistem komputer.
b. Perangkat lunak sistem bantuan (*utility*), yaitu program yang ditulis untuk bantuan yang berhubungan dengan sistem komputer, misalnya memformat disk, menyalin *disk*, mencegah dan membersihkan virus dan lain sebagainya.
c. Perangkat lunak bahasa (*language software*), yaitu program yang digunakan untuk menerjemahkan instruksi-instruksi yang ditulis dalam bahasa pemrograman ke dalam bahasa mesin supaya dapat dimengerti oleh komputer.

2. Perangkat lunak aplikasi (*application software*), yaitu program yang ditulis dan diterjemahkan oleh *language software* untuk menyelesaikan suatu aplikasi tertentu.

2.4 Siklus Hidup Pengembangan Sistem

Siklus Hidup Pengembangan Sistem (SHPS) atau *Software Development Life Cycle* (SDLC) adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik) (Sommerville dan Sawyer, 1997).

2.4.1 Tahapan SDLC

SDLC memiliki tahapan-tahapan dalam pengembangan sistem. Tahapan tersebut, yaitu *software requirement*, *software design*, *software construction*, *software testing*, *software maintenance*.

A. *Software Requirement*

Software requirement memiliki beberapa tahapan, yaitu:

A.1 Elisitasi Kebutuhan

Elisitasi atau pengumpulan kebutuhan merupakan aktivitas awal dalam proses rekayasa perangkat kebutuhan. Sebelum kebutuhan dapat dianalisis, dimodelkan, atau ditetapkan, kebutuhan harus dikumpulkan melalui proses elisitasi. Elisitasi kebutuhan adalah sekumpulan aktivitas yang ditunjukan untuk menemukan kebutuhan suatu sistem melalui komunikasi dengan pelanggan, pengguna sistem, dan pihak lain yang memiliki kepentingan dalam pengembangan sistem.

Sejalan dengan proses rekayasa kebutuhan secara keseluruhan, elisitasi kebutuhan bertujuan untuk:

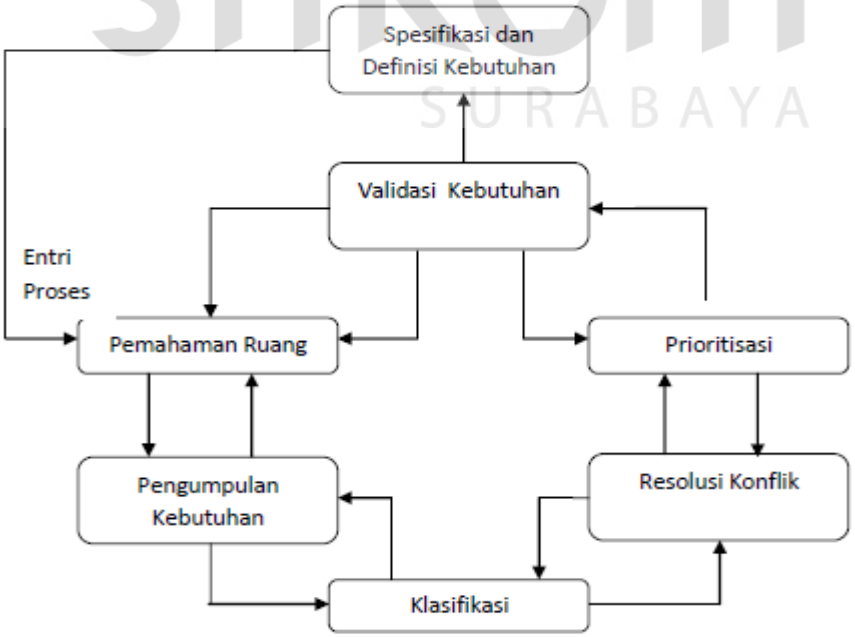
a. Mengetahui masalah apa saja yang perlu dipecahkan dan mengenali batasan-batasan sistem. Proses-proses dalam pengembangan perangkat lunak sangat ditentukan oleh seberapa dalam dan luas pengetahuan developer tentang permasalahan.
b. Mengenali siapa saja *stakeholder*, yaitu setiap pihak yang memiliki kepentingan terhadap sesuatu, dimana dalam konteks perangkat lunak adalah proyek pengembangan perangkat lunak itu sendiri. Beberapa yang dapat dikatakan sebagai *stakeholder* antara lain adalah konsumen atau *client* yang

membayar sistem, pengembang yang merancang, membangun, dan merawat sistem, dan pengguna yang berinteraksi dengan sistem untuk mendapatkan hasil kerja pengguna.

c. Mengenali tujuan dari sistem, yaitu sasaran-sasaran yang harus dicapai. Tujuan merupakan sasaran sistem yang harus dipenuhi, penggalian *high level goals* di awal proses pengembangan sangatlah penting karena bertujuan lebih terfokus pada ranah masalah dan kebutuhan *stakeholder* daripada solusi yang dimungkinkan untuk masalah tersebut.

A.2 Analisis

Tahap analisis merupakan tahap identifikasi, seleksi, dan perencanaan sistem yang bertujuan untuk mendeteksi dan memberikan solusi antar kebutuhan serta mengetahui ruang lingkup perangkat lunak dan bagaimana perangkat lunak tersebut berinteraksi dengan lingkungan.



Gambar 2.2 Tahapan Analisis Kebutuhan (England dkk, 2004)

Diagram pada Gambar 2.2 menunjukkan tahapan-tahapan di dalam analisis kebutuhan. Pada dasarnya, aktivitas analisis dibutuhkan dalam setiap proses dalam daur hidup pengembangan perangkat lunak.

A.3 Spesifikasi Kebutuhan

Spesifikasi kebutuhan yang dibutuhkan dalam pengembangan sistem, yaitu:

a. Kebutuhan Fungsional

Kebutuhan fungsional merupakan dasar dari penyusunan fungsi-fungsi yang akan dibangun di dalam perangkat lunak. Fungsi-fungsi perangkat lunak tersebut telah melewati proses identifikasi kebutuhan setiap pengguna, sehingga diharapkan fungsi-fungsi tersebut akan tepat pada sasaran dan sesuai dengan apa yang dibutuhkan.

b. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah fungsi-fungsi di luar fungsi utama yang mendukung fungsi utama itu sendiri. Adapun beberapa fungsi tersebut menurut ISO 9001, yaitu *time behavior*, *security*, *operability*, *accuracy*, dan *maintain ability*.

A.4 *Requirement Verification and Validation*

Dokumen *Software Requirement Specification* (SRS) dibentuk dan disusun berdasarkan data-data yang telah diperoleh dari *client*. Data-data tersebut telah melewati proses seleksi dan analisis sehingga hasil yang didapat lebih spesifik dan lebih sesuai dengan permintaan *client*. Dokumen SRS harus diverifikasi kembali kepada *client* dengan tujuan agar aplikasi yang dibangun

benar-benar terarah dan tidak mengalami banyak perubahan yang signifikan sehingga secara tidak langsung akan meningkatkan efisiensi waktu pengerjaan. Setelah diverifikasi, *client* diharapkan dapat memberikan validasi terhadap spesifikasi kebutuhan perangkat lunak yang akan dibangun tersebut sebagai tanda kesepakatan antar kedua belah pihak sekaligus sebagai awal untuk melanjutkan ke tahap *Software Construction*.

B. *Software Design*

Tahap desain adalah tahapan merancang pemodelan data yang dapat divisualisasikan melalui *Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM). Pemodelan proses dapat divisualisasikan melalui *Data Flow Diagram* (DFD) atau melalui *Unified Modeling Language* (UML). Dalam tahap ini juga mentransformasikan hasil dari analisis kebutuhan menjadi kebutuhan yang sudah lengkap yang difokuskan pada bagaimana memenuhi fungsi-fungsi yang dibutuhkan. Desain tersebut mencakup desain *form* dan laporan, desain antarmuka dan dialog, desain basis data dan *file* (*framework*), dan desain proses atau desain struktur proses serta juga termasuk *flowchart* program. *Flowchart* adalah suatu penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. Keberadaan *flowchart* sangat membantu analis sistem dan *programmer* dalam memecahkan suatu permasalahan menjadi segmen-segmen yang lebih kecil sehingga mempermudah dalam melakukan analisis alternatif-alternatif lain dalam pengoperasian.

C. *Software Construction*

Tahap ini melakukan konversi hasil desain ke sistem informasi yang lengkap melalui tahapan *coding* atau pengodean termasuk bagaimana membuat basis data dan menyiapkan prosedur kasus pengujian, mempersiapkan berkas atau *file* pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program serta melakukan peninjauan pengujian. *Construction* ini memiliki beberapa tahapan secara umum, yaitu:

a. *Software Construction Fundamentals*

Pada tahap pertama, dilakukan pendefinisian dasar tentang prinsip-prinsip yang digunakan dalam proses implementasi seperti minimalisasi kompleksitas, mengantisipasi perubahan, dan standar yang digunakan.

b. *Managing Construction*

Bagian ini mendefinisikan tentang model implementasi yang digunakan, rencana implementasi, dan ukuran pencapaian dari implementasi tersebut.

c. *Practical Considerations*

Bagian ini membahas tentang desain implementasi yang digunakan, bahasa pemrograman yang digunakan, kualitas dari implementasi yang dilakukan, proses pengujian, dan integrasi.

D. *Software Testing*

Tahap ini mendemonstrasikan sistem perangkat lunak yang telah selesai dibuat untuk dijalankan, apakah telah sesuai dengan kebutuhan yang telah ditentukan sebelumnya dan dapat diadaptasi pada lingkungan sistem yang baru. Tahapan ini tertuang dalam suatu dokumen *Test Plan*, yang dimulai dari membuat *Software Testing Fundamentals* yang berisi tentang penjelasan penting mengenai

terminology testing, kemudian selanjutnya merancang *Test Levels* yang terbagi antara target pengujian dan objektif dari pengujian. Pada tahap berikutnya adalah mendefinisikan *Test Techniques*, yaitu tentang bagaimana teknik yang digunakan termasuk dasar-dasar pengujian berdasarkan intuisi dan pengalaman serta teknik pengujian secara teknik *coding*, teknik kesalahan, teknik penggunaan, dan teknik terkait lainnya. Tahap selanjutnya adalah mendefinisikan *Test-Related Measures*, yaitu ukuran-ukuran pencapaian *testing* yang telah dilakukan untuk kemudian dievaluasi kembali. Tahap terakhir adalah mendefinisikan *Test Process* yang berisi tentang aktifitas pengujian.

E. *Software Maintenance*

Tahap ini adalah tahap yang mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan pengguna termasuk implementasi akhir dan proses peninjauan kembali. Pemeliharaan sistem ini terdiri dari beberapa jenis, yaitu: a) *Corrective*, yaitu memperbaiki desain dan *error* pada program; b) *Adaptive*, yaitu memodifikasi sistem untuk beradaptasi dengan perubahan lingkungan; c) *Perfective*, yaitu melibatkan sistem untuk menyelesaikan masalah baru atau mengambil kesempatan untuk penambahan fitur; d) *Preventive*, yaitu menjaga sistem dari kemungkinan masalah pada masa yang akan datang.

Prosedur pemeliharaan tersebut disusun dalam beberapa tahapan. Tahap awal adalah menyusun *Software Maintenance Fundamentals* yang berisi tentang dasar-dasar pemeliharaan, segala yang dibutuhkan untuk melakukan pemeliharaan, dan kategori pemeliharaan. Selanjutnya adalah mendefinisikan *Key Issue in Software Maintenance*, yang berisi tentang teknik pemeliharaan,

manajemen pemeliharaan dan biaya, serta ukuran pemeliharaan perangkat lunak. Tahap selanjutnya adalah mendefinisikan proses dan aktivitas pemeliharaan tersebut ke dalam *Maintenance Process*.

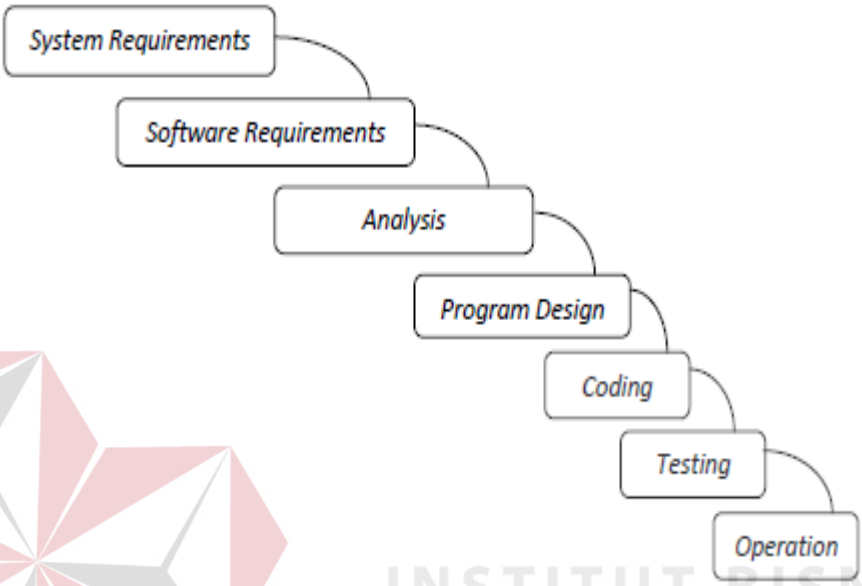
2.4.2 Model SDLC

SDLC memiliki beberapa model dalam penerapan tahapan prosesnya. Masing-masing model memiliki kelemahan dan kelebihan, sehingga hal yang terpenting adalah mengenali tipe pelanggan dan memilih menggunakan model SDLC yang sesuai dengan karakter pelanggan dan sesuai dengan karakter pengembang perangkat lunak. Model SDLC yang digunakan pada penelitian ini adalah Model *Waterfall*.

Model SDLC *Waterfall* sering juga disebut model sekuensial linier atau siklus hidup klasik. Model *Waterfall* menyediakan pendekatan siklus hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung. Dari kenyataan yang terjadi sangat jarang Model *Waterfall* dapat dilakukan sesuai siklusnya karena seperti perubahan spesifikasi perangkat lunak terjadi di tengah siklus pengembangan, adanya kesulitan bagi pelanggan untuk mendefinisikan semua spesifikasi di awal siklus pengembangan.

Pelanggan sering kali membutuhkan contoh untuk menjabarkan spesifikasi kebutuhan sistem lebih lanjut, serta pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan di akhir siklus pengembangan. Dengan berbagai kelemahan yang dimiliki Model *Waterfall* namun model ini telah menjadi dasar dari model-model lain dalam melakukan perbaikan model pengembangan perangkat lunak.

Model *Waterfall* ini adalah model SDLC yang paling sederhana, dan hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah. Tahapan dari Model *Waterfall* ini dapat dilihat pada Gambar 2.3.



Gambar 2.3 Model *Waterfall* (Royce, 1970)

Berikut ini akan dijelaskan secara singkat tentang tahapan dalam Model *Waterfall*, yaitu:

1. *System Requirements*

   Merupakan tahap pengumpulan data tentang kondisi awal dari suatu permasalahan yang akan diselesaikan. Data tersebut seperti siapa saja *stakeholder* yang ada, bagaimana keadaan sistem yang sedang digunakan saat ini dan perubahan seperti apa yang diinginkan oleh para *stakeholder* tersebut.

2. *Software Requirements*

   Tahap selanjutnya adalah mendefinisikan kebutuhan perangkat lunak yang akan dibangun sesuai dengan apa yang diinginkan oleh para *stakeholder*.

3. *Analysis*

Tahap ini merupakan tahap mengidentifikasi, menyeleksi, dan merencanakan sistem yang bertujuan untuk mendeteksi dan memberikan solusi terhadap permasalahan yang ada.

4. *Program Design*

   Tahap ini melakukan desain, pendefinisian dan pengolahan data yang terkait dengan fungsi, desain basis data, pendefinisian pengolahan basis data, waktu eksekusi, mendefinisikan antarmuka dan penjelasan tentang *input*, *process*, dan *output*.

5. *Coding*

   Tahap untuk melakukan pengodean untuk membangun perangkat lunak sesuai dengan hasil dari desain program sekaligus menyiapkan dokumentasi untuk setiap aktivitas pengodean.

6. *Testing*

   Melakukan uji kelayakan perangkat lunak yang telah dibangun sesuai dengan skenario dan *test plan* yang disiapkan.

7. *Operations*

   Tahap ini adalah pengimplementasian dan instalasi perangkat lunak. Perangkat lunak tersebut akan diadaptasi dengan sistem yang lama untuk kemudian dilakukan evaluasi.

2.5 Persediaan

Persediaan adalah sumber daya yang menunggu proses lebih lanjut (*idle resource*). Yang dimaksud dengan proses lebih lanjut tersebut adalah berupa kegiatan produksi pada sistem manufaktur, kegiatan pemasaran pada sistem

distribusi ataupun kegiatan konsumsi pangan pada sistem rumah tangga (Nasution, 1999).

Timbulnya persediaan dalam suatu sistem, baik sistem manufaktur maupun non manufaktur adalah merupakan akibat dari 3 kondisi sebagai berikut (Nasution, 1999):

1. Mekanisme pemenuhan atas permintaan (*transaction motive*). Permintaan akan suatu barang tidak akan dapat dipenuhi dengan segera bila barang tersebut tidak tersedia sebelumnya, karena untuk mengadakan barang dibutuhkan waktu untuk pembuatannya maupun untuk mendatangkannya. Hal ini berarti bahwa adanya persediaan merupakan hal yang sulit dihindarkan.
2. Adanya keinginan untuk mencegah ketidakpastian (*precautionary motive*). Ketidakpastian yang dimaksudkan adalah:
   a. Adanya permintaan yang bervariasi dan tidak pasti dalam jumlah maupun waktu kedatangan.
   b. Waktu pembuatan yang cenderung tidak konstan antara satu produk dengan produk yang lain.
   c. Waktu tunggu (*lead time*) yang cenderung tidak pasti karena berbagai faktor yang tak dapat dikendalikan sepenuhnya.

   Ketidakpastian ini akan dicegah oleh jenis persediaan yang disebut persediaan yang harus ada (*safety stock*). Persediaan yang harus ada ini digunakan jika permintaan melebihi peramalan, produksi lebih rendah dari rencana, atau waktu tunggu (*lead time*) lebih panjang dari yang diperkirakan semula.

3. Keinginan melakukan spekulasi (*speculative motive*) yang bertujuan mendapatkan keuntungan besar dari kenaikan harga barang di masa mendatang.

Dari uraian di atas, dapat ditarik kesimpulan bahwa fungsi utama persediaan adalah menjamin kelancaran mekanisme pemenuhan permintaan barang sesuai dengan kebutuhan konsumen sehingga sistem yang dikelola dapat mencapai kinerja (*performance*) yang optimal.

2.6 Peramalan

Definisi peramalan sendiri sebenarnya beragam. Berikut beberapa definisi tentang peramalan (Santoso, 2009):

1. Perkiraan munculnya sebuah kejadian di masa depan berdasarkan data yang ada di masa lampau.
2. Proses menganalisis data historis dan data saat ini untuk menentukan pola di masa mendatang.
3. Proses estimasi dalam situasi yang tidak diketahui.
4. Pernyataan yang dibuat tentang masa depan.
5. Penggunaan ilmu dan teknologi untuk memperkirakan situasi di masa depan.
6. Upaya sistematis untuk mengantisipasi kejadian atau kondisi di masa depan.

Dari beberapa definisi di atas, dapat disimpulkan bahwa peramalan berkaitan dengan upaya memperkirakan apa yang terjadi di masa depan, berbasis pada metode ilmiah (ilmu dan teknologi) serta dilakukan secara sistematis. Walaupun demikian, kegiatan peramalan tidaklah semata-mata berdasarkan prosedur ilmiah atau terorganisir, karena ada kegiatan peramalan yang menggunakan intuisi (perasaan) atau lewat diskusi informal dalam sebuah grup.

Tabel 2.1 Ciri Sebuah Kegiatan Peramalan (Santoso, 2009)

Aspek	Peramalan
Fokus	Data di masa lalu
Tujuan	Menguji perkembangan saat ini dan relevansinya di masa mendatang
Metode	Proyeksi berdasar ilmu statistik, diskusi, dan review program
Orang yang terlibat	Pembuat keputusan, petugas administrasi, praktisi, analis
Frekuensi	Regular (teratur)
Kriteria Keberhasilan	Tidak sekedar akurasi, namun bersifat pembelajaran

Dari kriteria di atas, terlihat bahwa peramalan adalah kegiatan yang bersifat teratur, berupaya memprediksi masa depan dengan menggunakan tidak hanya metode ilmiah, namun juga mempertimbangkan hal-hal yang bersifat kualitatif, seperti perasaan, pengalaman seseorang dan lainnya (Santoso, 2009).

2.7 Tahapan Peramalan

Menurut Santoso (2009), untuk mendapatkan hasil peramalan yang baik dan secara efektif dapat menjawab masalah yang ada, kegiatan peramalan sebaiknya mengikuti tahapan baku berikut ini:

1. Perumusan masalah dan pengumpulan data

   Tahap pertama yang sebenarnya penting dan menentukan keberhasilan peramalan adalah menentukan masalah tentang apa yang akan diprediksi. Formulasi masalah yang jelas akan menuntun pada ketepatan jenis dan banyaknya data yang akan dikumpulkan. Dapat saja masalah telah ditetapkan, namun data yang relevan tidak tersedia. Hal ini akan memaksa diadakannya perumusan ulang atau mengubah metode peramalan.

2. Persiapan data

Setelah masalah dirumuskan dan data telah terkumpul, tahap selanjutnya adalah menyiapkan data hingga dapat diproses dengan benar. Hal ini diperlukan, karena dalam praktek ada beberapa masalah berkaitan dengan data yang telah terkumpul:

a. Jumlah data terlalu banyak
b. Jumlah data justru terlalu sedikit
c. Data harus diproses terlebih dahulu
d. Data tersedia namun rentang waktu data tidak sesuai dengan masalah yang ada
e. Data tersedia namun cukup banyak data yang hilang (*missing*), yakni data yang tidak lengkap.

3. Membangun model

   Setelah data dianggap memadai dan siap dilakukan kegiatan prediksi, proses selanjutnya adalah memilih metode (model) yang tepat untuk melakukan peramalan pada data tersebut.

4. Implementasi model

   Setelah metode peramalan ditetapkan, maka model dapat diterapkan pada data dan dapat dilakukan prediksi pada data untuk beberapa periode ke depan.

5. Evaluasi peramalan

   Hasil peramalan yang telah ada kemudian dibandingkan dengan data aktual. Metode peramalan tidak dapat memprediksi data di masa depan secara tepat yang ada adalah ketepatan prediksi. Untuk itu, pengukuran kesalahan peramalan dilakukan untuk melihat apakah metode yang telah digunakan sudah memadai untuk memprediksi sebuah data.

2.8 Jenis Data pada Kegiatan Peramalan

Data yang akan diprediksi secara umum dapat dibagi menjadi dua tipe, yakni data kualitatif dan data kuantitatif. Menurut Arsyad (2001), data kualitatif adalah serangkaian observasi yang setiap observasinya terdapat dalam sampel (atau populasi) tergolong pada salah satu dari kelas-kelas yang eksklusif secara bersama dan kemungkinannya tidak dapat dinyatakan dalam angka. Sebaliknya data kuantitatif adalah serangkaian observasi yang dapat dinyatakan dengan angka.

Data kuantitatif dapat dibagi menjadi dua bagian, yaitu (Santoso, 2009):

1. Data *time series*

   Data *time series* adalah data yang ditampilkan berdasarkan waktu, seperti data bulanan, data harian, data mingguan atau jenis waktu yang lain. Contoh data *time series* adalah data penjualan bulanan motor di daerah A dari tahun 2000 sampai 2007.

2. Data *cross-sectional*

   Data *cross-sectional* adalah data yang tidak berdasar waktu tertentu, namun pada satu (titik) waktu tertentu. Contoh data *cross-sectional* adalah data biaya promosi di sepuluh area pemasaran produk X selama bulan Januari 2008.

2.9 Data Stasioner dan Data Tidak Stasioner

Data stasioner adalah data yang rata-rata nilainya tidak berubah dari waktu ke waktu, atau dapat dikatakan data bersifat stabil. Sebaliknya, data dapat saja tidak stasioner, ketika pada uji pola data didapati adanya *trend* atau pola musiman (pengaruh musim) (Santoso, 2009).

2.10 Uji Pola Data

Uji pola data pada intinya adalah menguji apakah sebuah data dapat dikatakan stasioner ataukah tidak. Jika pada data terdapat *trend* atau ada komponen musiman atau siklis, dikatakan bahwa data tidak dapat dikatakan stasioner. Namun sebaliknya, jika pada data tidak ada *trend*, musiman ataukah siklis, maka data dapat dikatakan stasioner. Stasioneritas data penting untuk menentukan lebih jauh metode peramalan apa yang tepat dilakukan. Metode untuk data yang stasioner akan berbeda dengan metode peramalan untuk data yang tidak stasioner (Santoso, 2009).

Salah satu ciri data stasioner adalah adanya korelasi antar data penjualan. Otokorelasi dapat membuktikan adanya korelasi antar data penjualan. Pada umumnya, jika sebuah data saling berkorelasi pada jarak waktu yang berdekatan, misalnya antara waktu t dengan waktu sebelumnya (t-1), maka dikatakan data mempunyai kecenderungan berotokorelasi. Besaran korelasi antara data ke t dan data ke t-1 cukup tinggi, kemudian menurun secara bertahap. Data demikian bisa diduga mempunyai unsur *trend* di dalamnya dan tidak bersifat acak. Sebaliknya, data yang mempunyai korelasi antar waktu yang rendah serta tidak menunjukkan pola penurunan otokorelasi yang bertahap, pada data tersebut dapat dikatakan tidak ada unsur *trend* (Santoso, 2009).

Cara lain yang cukup praktis adalah melihat tampilan grafik data. Jika ada unsur *trend*, maka terlihat data dari waktu ke waktu cenderung semakin naik atau semakin turun. Sebaliknya, jika data stasioner, grafik akan relatif tidak menaik atau menurun. Ada pola kenaikan data yang diikuti oleh pola data yang menurun (Santoso, 2009).

Pengujian stasioneritas data dapat dilakukan dengan dua cara yaitu dengan grafik (lebih praktis) atau dengan menghitung otokorelasi. Sebaliknya kedua cara dilakukan secara bersama-sama karena saling melengkapi (Santoso, 2009).

Untuk menghitung otokorelasi digunakan persamaan sebagai berikut (Santoso, 2009):

$$r_k = \frac{\sum_{t=k+1}^{n}(Y_t-\bar{Y})(Y_{t-k}-\bar{Y})}{\sum_{t=1}^{n}(Y_t-\bar{Y})^2} \quad \text{..... 2.1}$$

dengan:

r_k = koefisien otokorelasi

Y_t = data saat ini (yang ke t)

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

Y_{t-k} = data pada periode k sebelum data saat ini. Jika k=1, maka menjadi Y_{t-1} atau data satu lag dari periode sebelumnya. Jika k=2, maka menjadi Y_{t-2} atau data dua lag dari periode awal.

n = jumlah data

Setelah nilai otokorelasi didapat, maka langkah selanjutnya menguji nilai otokorelasi dengan menggunakan uji t. Proses pengujian dengan menggunakan t (Santoso, 2009):

1. Merumuskan hipotesis, yang secara standar dapat dinyatakan sebagai berikut:

   H0: 1 = 0, atau koefisien korelasi yang didapat tidak signifikan.

   H1: 1 ≠ 0, atau koefisien korelasi yang didapat memang nyata.

2. Menghitung t hitung dan t tabel.

   t hitung didapat dengan persamaan:

$$t = \frac{r_1}{SE(r_1)} \quad \text{.....} \quad 2.2$$

dengan

$$SE = \frac{\sqrt{1+2\sum_{i=1}^{k-1} r_i^2}}{n} \quad \text{.....} \quad 2.3$$

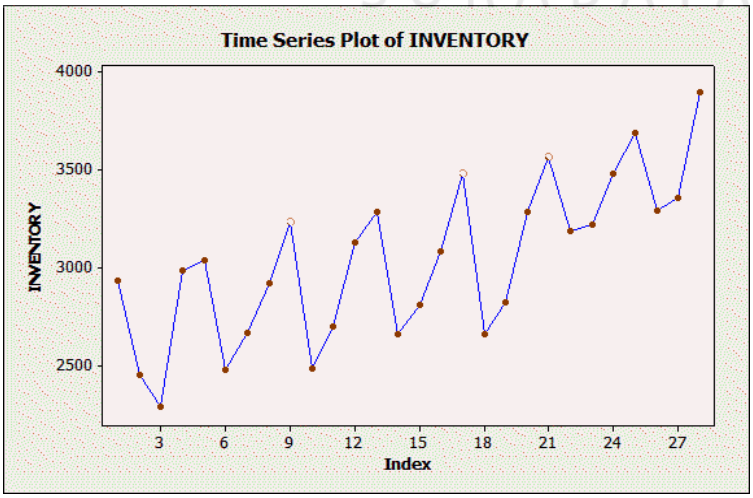
t tabel dihitung dari tabel t pada tingkat kepercayaan tertentu (biasanya 95%) dan dengan df = n – 1.

3. Pengambilan keputusan:

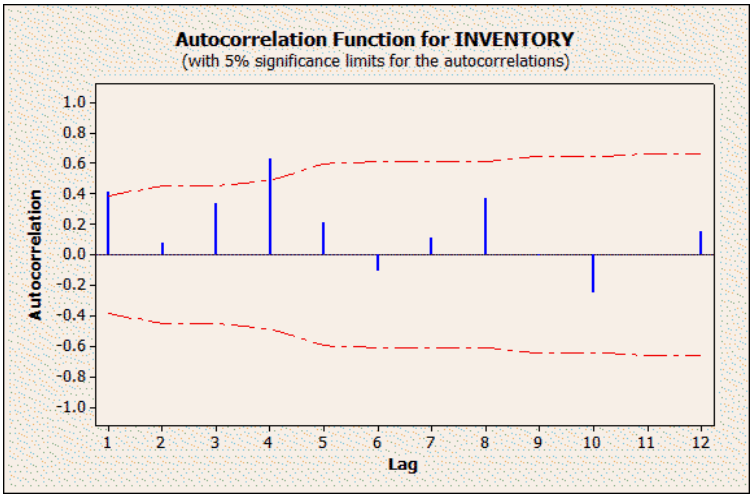
   Jika t hitung < t tabel, H0 diterima.

   Jika t hitung > t tabel, H0.

Cara kedua pengujian stasioneritas dengan melihat pola data secara grafis dapat dilakukan dengan menggunakan *software* Minitab 16. Apabila terdapat pola data yang naik dan turun seperti pada Gambar 2.4, maka data dikatakan tidak stasioner karena adanya *trend*. Selain data itu dikatakan stasioner karena adanya *trend*, diperkirakan data tersebut terdapat pola data musiman. Untuk memastikan data tersebut terdapat pola data musiman, maka dapat dilakukan uji otokorelasi.



Gambar 2.4 *Time Series*



Gambar 2.5 Otokorelasi

Hasil uji otokorelasi yang ditunjukkan pada Gambar 2.5 menunjukkan adanya korelasi. Korelasi tersebut disimbolkan dengan bar berwarna biru pada lag pertama sampai dengan lag kedua belas. Lag pertama dan lag ketiga yang melewati garis batas merah menunjukkan bahwa terdapat *trend* dan data tidak stasioner. Bar lag pertama sampai keempat menunjukkan bar pertama dan bar keempat lebih tinggi dibandingkan dengan bar kedua dan bar ketiga. Perbedaan ketinggian untuk setiap empat lag tersebut membuktikan adanya pengaruh musiman selain adanya *trend*. Nilai korelasi dapat dilihat pada Gambar 2.6.


```
Welcome to Minitab, press F1 for help.

Results for: INVENTORY SEASONAL.MTW

Time Series Plot of INVENTORY


Autocorrelation Function: INVENTORY

Lag         ACF       T     LBQ
  1    0.413680    2.19    5.32
  2    0.071963    0.33    5.49
  3    0.337119    1.53    9.31
  4    0.624801    2.63   22.97
  5    0.207303    0.71   24.54
  6   -0.112104   -0.38   25.02
  7    0.111980    0.38   25.52
  8    0.368509    1.23   31.23
  9   -0.011761   -0.04   31.23
 10   -0.253196   -0.81   34.23
 11   -0.003678   -0.01   34.23
 12    0.149197    0.46   35.39
```

Gambar 2.6 Nilai Otokorelasi

2.11 Pemulusan Eksponensial Winter

Salah satu bentuk pemulusan lain yang berguna dikembangkan oleh Winters pada awal dekade 1960-an. Metode ini memberikan hasil yang serupa dengan pemulusan eksponensial linier, tetapi memiliki manfaat tambahan dalam kemampuannya untuk menangani data musiman selain data yang memiliki *trend* (Makridakis dan Wheelwright, 1992).

Pemulusan eksponensial linier dan musiman Winters didasari oleh tiga persamaan, yang masing-masing memuluskan satu faktor yang berkaitan dengan satu di antara tiga komponen pola - faktor *random*, *trend*, dan musiman. Dalam hal ini, metode ini serupa dengan pemulusan eksponensial linier, yang memuluskan faktor *random* dan menyesuaikan dengan *trend*. Tetapi, Metode Winters mencakup parameter tambahan untuk menangani faktor musiman.

Keempat persamaan yang digunakan dalam Metode Winter adalah sebagai berikut (Arsyad, 1994):

1. Pemulusan Eksponensial:

$$A_t = \alpha \frac{Y_t}{S_{t-L}} + (1-\alpha)(A_{t-1} + T_{t-1}) \quad \text{..... 2.4}$$

2. Estimasi *Trend*:

$$T_t = \beta(A_t - A_{t-1}) + (1-\beta)T_{t-1} \quad \text{..... 2.5}$$

3. Estimasi Musiman:

$$S_t = \gamma \frac{Y_t}{A_t} + (1-\gamma)S_{t-L} \quad \text{..... 2.6}$$

4. Ramalan pada periode *p* di masa datang

$$\hat{Y}_{t+p} = (A_t + pT_t)S_{t-L+p} \quad \text{..... 2.7}$$

dengan:

A_t = nilai pemulusan yang baru

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

Y_t = data yang baru atau yang sebenarnya pada periode *t*

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

T_t = estimasi *trend*

μ = konstanta pemulusan untuk estimasi musiman ($0 \leq \mu \leq 1$)

S_t = estimasi musiman

p = periode yang diramalkan

L = panjangnya musim

$\hat{Y}_{t+p}$ = ramalan pada periode p

t = waktu

2.12 Alat Ukur Kesalahan Prediksi

Oleh karena teknik peramalan kuantitatif biasanya menggunakan data runtut waktu, maka notasi matematis harus digunakan untuk menunjukkan suatu periode waktu tertentu. Huruf *Y* digunakan untuk menunjukkan suatu variabel data runtut waktu. Periode waktu dari satu variabel ditunjukkan sebagai subskrip. Oleh karena itu, Y_t menunjukkan nilai *Y* pada periode *t* (Arsyad, 1994).

Notasi matematis juga harus digunakan untuk membedakan nilai variabel data runtut waktu sebenarnya dengan nilai peramalan. Tanda "^" (topi) diletakkan di atas variabel yang sedang diramalkan. Nilai peramalan untuk Y_t adalah $\hat{Y}_t$. Akurasi dari teknik peramalan sering kali dinilai dengan cara membandingkan data asli yakni Y_1, Y_2, ... dengan nilai-nilai data hasil peramalan $\hat{Y}_1$, $\hat{Y}_2$, ...

Notasi dasar peramalan adalah sebagai berikut:

Y_t = nilai data runtut waktu periode *t*

$\widehat{Y_t}$ = nilai peramalan dari Y_t

$e_t = Y_t - \widehat{Y_t}$ = residual atau kesalahan peramalan

Beberapa metode telah digunakan untuk menunjukkan kesalahan yang disebabkan oleh suatu teknik peramalan tertentu. Hampir semua ukuran tersebut menggunakan hasil rata-rata beberapa fungsi dari perbedaan antara nilai sebenarnya dengan nilai peramalannya. Perbedaan antara nilai sebenarnya dengan nilai peramalan ini biasanya disebut sebagai *residual*.

Persamaan 2.8 digunakan untuk menghitung kesalahan atau residual dari setiap periode peramalan.

$$e_t = Y_t - \widehat{Y_t} \quad \text{.....} \quad 2.8$$

dengan:

e_t = kesalahan peramalan pada periode *t*

Y_t = nilai sebenarnya pada periode *t*

$\widehat{Y_t}$ = nilai peramalan pada periode *t*

Salah satu cara untuk mengevaluasi teknik peramalan adalah menggunakan penjumlahan kesalahan absolut. Simpangan absolut rata-rata atau *Mean Absolute Deviation* (MAD) mengukur akurasi peramalan dengan merata-ratakan kesalahan peramalan (nilai absolutnya). MAD ini sangat berguna jika seorang analis ingin mengukur kesalahan peramalan dalam unit ukuran yang sama seperti data aslinya. Persamaan 2.9 menunjukkan bagaimana cara menghitung MAD (Arsyad, 1994: 58).

$$\text{MAD} = \frac{\sum_{t=1}^{n}(Y_t - \widehat{Y_t})}{n} \quad \text{.....} \quad 2.9$$

Kesalahan rata-rata kuadrat atau *Mean Squared Error* (MSE) merupakan metode alternatif dalam mengevaluasi suatu teknik peramalan. Setiap kesalahan

atau residual dikuadratkan, kemudian dijumlahkan dan dibagi dengan jumlah observasi. Pendekatan ini mengakibatkan suatu kesalahan peramalan yang besar karena dikuadratkan. Pendekatan ini penting karena menghasilkan kesalahan yang lebih sesuai untuk hasil peramalan yang biasanya menghasilkan kesalahan yang lebih kecil dan kadang-kadang menghasilkan kesalahan yang sangat besar. Persamaan 2.10 menunjukkan bagaimana cara menghitung MSE.

$$\text{MSE} = \frac{\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2}{n} \quad \text{.....} \quad 2.10$$

Adakalanya lebih bermanfaat jika kesalahan peramalan dihitung secara persentase dibandingkan secara absolut. Rata-rata persentase kesalahan absolut atau *Mean Absolute Percentage Error* (MAPE) dihitung dengan menemukan kesalahan absolut setiap periode, kemudian menghitung kesalahan absolut pada periode tersebut, dan akhirnya merata-rata persentase kesalahan absolut ini. Pendekatan ini sangat berguna jika ukuran variabel peramalan merupakan faktor penting dalam mengevaluasi akurasi peramalan tersebut. MAPE memberikan petunjuk seberapa besar kesalahan peramalan dibandingkan dengan nilai sebenarnya dari series tersebut. MAPE juga dapat digunakan untuk memperbandingkan akurasi dari teknik yang sama atau berbeda pada dua series yang berbeda. Persamaan 2.11 menunjukkan bagaimana cara menghitung MAPE (Arsyad, 1994).

$$\text{MAPE} = \frac{\sum_{t=1}^{n}\frac{|Y_t - \hat{Y}_t|}{Y_t}}{n} \quad \text{.....} \quad 2.11$$

Sering kali perlu juga untuk menentukan apakah suatu metode peramalan bias atau tidak (secara konsisten tinggi atau rendah). Rata-rata persentase kesalahan atau *Mean Percentage Error* (MPE) digunakan dalam kasus seperti ini.

MPE dihitung dengan cara menemukan kesalahan setiap periode, kemudian menghitung persentase kesalahan pada periode tersebut, dan kemudian merata-rata persentase kesalahan tersebut. Jika pendekatan peramalan tersebut tidak bias, maka persamaan 2.12 akan menghasilkan persentase mendekati nol. Jika hasil persentase negatifnya cukup besar, maka metode peramalan tersebut menghasilkan hasil ramalan yang terlalu tinggi, demikian sebaliknya.

$$\text{MPE} = \frac{\sum_{t=1}^{n} \frac{|Y_t - \widehat{Y_t}|}{Y_t}}{n} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.12$$

Empat cara pengukuran akurasi peramalan yang dibahas sebelumnya digunakan untuk tujuan berikut (Arsyad, 1994):

1. Pembandingan akurasi dari dua teknik peramalan yang berbeda.
2. Pengukuran kegunaan atau reliabilitas suatu teknik peramalan.
3. Pencarian teknik peramalan yang optimal.

2.13 *Fixed Time Period System* (P Model)

Fixed Time Period System (FTPS) adalah suatu sistem cara pemesanan bahan yang jarak atau interval waktu dari pemesanan tetap tetapi dengan jumlah berbeda-beda, misalnya tiap minggu atau bulan (Chase dkk, 2006). Jarak waktu pemesanan yang tetap membuat pemesanan dilakukan tanpa memperhatikan jumlah persediaan yang masih ada. Banyaknya jumlah barang yang dipesan ditetapkan sebesar selisih dari jumlah persediaan maksimum yang telah ditentukan dengan jumlah persediaan yang tersisa.

Berikut ini adalah persamaan dari FTPS (P Model):

q = rata-rata permintaan + *safety* stok – persediaan saat ini

$$q = \bar{d}(T + L) + Z\sigma_{T+L} - I \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.13$$

dengan:

q = jumlah pesanan

T = periode *review*

L = *lead time*

$\bar{d}$ = rata-rata permintaan

z = standar deviasi

σ_{T+L} = standar devisiasi permintaan selama periode *review* dan *lead time*

I = *quantity on hand + quantity on order*

2.14 *Black Box Testing*

Menurut Rizky (2011), *black box testing* adalah tipe pengujian yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya. Para penguji memandang perangkat lunak seperti layaknya sebuah "kotak hitam" yang tidak penting dilihat isinya tapi cukup dikenai proses pengujian di bagian luar. Jenis pengujian ini hanya memandang perangkat lunak dari sisi spesifikasi dan kebutuhan yang telah didefinisikan pada saat awal perancangan.

Beberapa keuntungan yang diperoleh dari jenis pengujian ini antara lain:

1. Anggota tim penguji tidak harus dari seseorang yang memiliki kemampuan teknis di bidang pemrograman.
2. Kesalahan dari perangkat lunak ataupun *bug* sering ditemukan oleh komponen penguji yang berasal dari pengguna.
3. Hasil dari *black box testing* dapat memperjelas kontradiksi ataupun kerancuan yang mungkin timbul dari eksekusi sebuah perangkat lunak.
4. Proses pengujian dapat dilakukan lebih cepat dibandingkan *white box testing*.