BAB II

LANDASAN TEORI

2.1. Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju.

Menurut Whitten (dalam Kristanto, 1994 : 60) Perancangan Sistem adalah "Proses dimana keperluan pengguna dirubah ke dalam bentuk paket perangkat lunak dan atau kedalam spesifikasi pada komputer yang berdasarkan pada sistem informasi.".

2.2. *Cutting Stock Optimization*

Cutting stock optimization merupakan permasalahan optimasi dalam pengkombinasian, sehingga dapat ditentukan solusi dari beberapa solusi yang mungkin, yang memenuhi fungsi pembatas yang ada. Solusi yang ditawarkan adalah dengan mengkombinasikan beberapa pieces dengan ukuran berbeda ke dalam persegi empat (bahan baku) sehingga didapatkan sisa kaca seminimal mungkin.

2.2.1. Karakteristik Pemotongan Bahan (*Cutting Stock*)

Karakteristik pemotongan bahan adalah :

a. Terdapat bahan baku yang berbentuk persegi empat (selanjutnya disebut *rectangle*) yang mempunyai ukuran tertentu.

b. Terdapat m jenis potongan yang dihasilkan (yang selanjutnya disebut dengan *pieces*) yang masing-masing berukuran p dengan jumlah permintaan n tertentu.

c. Setiap potong mempunyai nilai tertentu yang bisa berupa keuntungan yang diperoleh atau berupa ukuran luas dalam upaya meminimasi sisa bahan baku.

d. Berusaha membentuk suatu *layout* potong yang meminimumkan fungsi tujuan yang melekat pada setiap potong yang ada.

2.2.2. Pola Pemotongan

a. *Guillotine Pattern*

Guillotine Pattern merupakan pola pemotongan yang dimulai dari satu sisi segi empat yang kemudian dilanjutkan pada sisi lainnya. Pemotongan pertama dengan tipe *Guillotine Pattern* adalah dengan memotong bahan baku dengan panjang atau lebar yang sama. Pemotongan tersebut menghasilkan dua atau lebih potongan yang mempunyai panjang atau lebar yang sama, bukan kedua-duanya.

b. *Non-guillotine Pattern*

Pemotongan dengan tipe *non-guillotine* dilakukan apabila ukuran *pieces* yang diinginkan tidak memungkinkan untuk digabung dengan *pieces* yang lain.

c. Pola Dua Tahap Pemotongan (*Two Stage Pattern*)

Tahap pertama, pemotongan secara paralel atau pemotongan bahan secara horizontal, sehingga *rectangle* terbagi menjadi beberapa rectangle dengan panjang yang sama. Tahap kedua adalah pemotongan satu persatu bagian *rectangle*.

d. Pola Tiga Tahap Pemotongan (*Three Stage Pattern*)

Tahap pertama, pemotongan *rectangle* menjadi bagian-bagian dengan panjang atau lebar yang sama. Arah pemotongan tersebut dapat secara vertikal maupun secara horizontal. Tahap kedua, hasil dari pemotongan tersebut dilanjutkan dengan

pemotongan satu persatu yang terlebih dahulu mengubah arah pemotongan. Tahap ketiga, pemotongan dilakukan pada bagian yang menghasilkan *pieces*.

2.3. Pembentukan Fungsi Tujuan

Sifat yang perlu diperhatikan dalam memilih kriteria untuk fungsi tujuan menurut Simatupang (1995 : 80-81) adalah sebagai berikut :

a. Lengkap

b. Operasional

c. Tidak Berlebihan

d. Minimum

2.3.1. Pengembangan Model Sistematis

Pengembangan model matematis dapat dimulai dengan menjawab ketiga pertanyaan berikut : Menurut Hamdy (1996 : 17)

a. Apa yang diusahakan untuk ditentukan oleh model tersebut? Dengan kata lain, apa variable (yang tidak diketahui) dari masalah tersebut?

b. Apa batasan yang harus dikenakan atas variable untuk memenuhi batasan sistem yang dimodel tersebut?

c. Apa tujuan (sasaran) yang harus dicapai untuk menentukan pemecahan optimum (terbaik) dari semua nilai yang layak dari variable tersebut?

2.3.2. Penambahan Batasan Baru

Penambahan batasan baru dapat menghasilkan satu di antara dua kondisi : Menurut Hamdy (1996 : 172)

a. Batasan itu dipenuhi oleh pemecahan saat ini, dalam kasus mana batasan tersebut berlebihan dan penambahannya tidak mengubah pemecahan.

b. Batasan tersebut tidak dipenuhi oleh pemecahan saat ini,. Dalam kasus ini pemecahan baru diperoleh dengan metode simpleks dual.

Yang kita lakukan disini adalah mendapatkan kembali kelayakan. Pertama – tama, tempatkan batasan baru tersebut dalam bentuk standar dengan menambahkan variabel slack atau surplus sebagaimana diperlukan. Lalu substitusi keluar setiap variabel dasar saat ini dalam batasan tersebut dalam bentuk variable non dasar (saat ini). Langkah terakhir adalah menambahkan batasan yag dimodifikasi ke tabel optimum saat ini dan menerapkan simpleks dual untuk memperoleh kembali kelayakan.

Batasan yang dimodifikasi ini sekarang ditambahkan ke tabel optimal saat ini seperti diberikan berikut ini:

Tabel 2.1 Kombinasi Potongan

Dasar	X1	X2	X3	X4	X5	X6	X7	Pemecahan
Z	0	0	1/3	4/3	0	0	0	38/3
X2	0	1	2/3	-1/3	0	0	0	4/3
X1	1	0	-1/3	2/3	0	0	0	10/3
X5	0	0	-1	1	1	0	0	3
X6	0	0	-2/3	1/3	0	1	0	2/3
X7	0	0	1/3	-2/3	0	0	1	-1/3

Dimana variabel :

X1 = Jumlah pola potong yang pertama.

X2 = Jumlah pola potong yang kedua.

X3 = Jumlah pola potong yang ketiga.

X4 = Jumlah pola potong yang keempat.

X5 = Jumlah pola potong yang kelima.

X6 = Jumlah pola potong yang keenam.

X7 = Jumlah pola potong yang ketujuh.

Jadi batasan baru tersebut yang diekspresikan dalam bentuk variabel nondasar menjadi:

(10/3) + (1/3)x3 – (2/3)x4 +x7 = 3

(1/3)x3 – (2/3)x4 + x7 = -1/3

2.3.3. Identifikasi Variabel

Dalam pemodelan, variabel yang teridentifikasikan hendaknya dapat digolongkan menjadi empat jenis yaitu : Menurut Simatupang (1995 : 94-95)

a. Variabel nominal

b. Variabel ordinal

c. Variabel interval

d. Variabel rasio

2.4. *Integer Linear Programming*

Linear Programming merupakan metode atau teknik matematik yang digunakan untuk membantu dalam pengambilan keputusan. Di dalam *linear programming*, seluruh fungsinya (fungsi objektif serta fungsi pembatas) haruslah linear.

Terdapat empat asumsi dasar dalam penyelesaian masalah dengan menggunakan model *linear programming*, yaitu : Menurut Lieberman dkk (1995 : 38-44)

a. *Proporsionality*.

b. *Divisibility*.

c. *Addivity.*

d. *Certainty* .

Integer Programming (IP) merupakan bentuk lain dari *Linear Programming* (LP) yang muncul karena tidak semua variabel keputusan dapat berupa bilangan pecahan dengan kata lain asumsi *divisibility* melemah atau hilang sama sekali.

Metode yang digunakan untuk memaksa pemecahan optimum dari *linear programming* yang dilonggarkan untuk bergerak ke arah pemecahan *integer* yang diinginkan adalah *branch & bound*. Algoritma *Branch & Bound* berlaku baik untuk masalah *integer* murni maupun masalah *integer* campuran. Keuntungan utamanya adalah bahwa batas atas tersebut dapat diestimasi dengan cepat dan dengan perhitungan minimal.

Tahapan yang dilakukan dalam algoritma *branch & bound* adalah sebagai berikut : Menurut Dimyati dkk (2003 : 217-227)

a. *Branching*

Apabila dari penyelesaian LP relaksasi diperoleh nilai variabel yang tidak *integer*, maka dilakukan *branching* atau pencabangan. Pencabangan dilakukan pada variabel yang bernilai pecahan atau tidak *integer*. Apabila terdapat lebih dari satu variabel yang bernilai pecahan, maka pilih secara sembarang (dari variabel pecahan tersebut) variabel yang akan dilakukan pencabangan.

b. *Bounding*

Setelah dilakukan *branching*, maka langkah selanjutnya adalah memilih salah satu subpersoalan yang belum diselesaikan dengan menerapkan aturan LIFO. Dari penghitungan yang dilakukan tersebut, diperoleh nilai *z* untuk masing-masing subpersoalan. Nilai *z* ini dijadikan *bound*.

c. *Fathoming*

Ada tiga situasi yang menyebabkan suatu subpersoalan *fathomed*, yaitu :

a. Apabila subpersoalan tersebut tidak *feasible*.

b. Apabila subpersoalan itu memberikan solusi optimal dimana seluruh variabelnya berharga *integer*.

c. Apabila nilai *z* optimal untuk subpersoalan itu tidak lebih baik dari nilai *z* optimal subpersoalan lain (dalam persoalan maksimasi berarti nilai *z* optimal dari subpersoalan itu tidak lebih besar daripada batas bawah yang telah diperoleh).

Apabila subpersoalan berada dalam situasi *a* atau *c* maka subpersoalan tersebut dapat diabaikan atau dieliminasi dari pertimbangan selanjutnya.

2.4.1 Metode Simpleks

Karena kesulitan menggambarkan grafik berdimensi banyak maka penyelesaian masalah pemrograman linier yang melibatkan lebih dari dua variable menjadi tidak praktis atau tidak mungkin. Dalam keadaan ini kebutuhan metode solusi yang lebih umum menjadi nyata. Metode umum ini dikenal dengan nama *Algoritma Simpleks* yang dirancang untuk menyelesaikan seluruh masalah program linier, baik yang melibatkan dua variable maupun lebih dua variabel.

Penyelesaian masalah pemrograman linear menggunakan metode simpleks ini melalui perhitungan ulang (*iteration*) dimana langkah langkah perhitungan yang sama diulang berkali-kali sebelum hasil optimum dicapai.

Dalam penyelesaian masalah program linear dengan grafik, telah dinyatakan bahwa solusi optimum selalu terletak pada titik pojok ruang solusi. Metode simpleks didasar pada gagasan ini, dengan langkah-langkah sebagai berikut :

a. Dimulai pada suatu titik pojok yang layak, biasanya titik asal (yang disebut sebagai solusi awal).
b. Bergerak dari suatu titik ke pojok yang lain yang berdekatan, pergerakan ini akan menghasilkan nilai fungsi tujuan yang lebih baik (meningkatkan untuk masalah maksimasi dan menurunkan untuk masalah minimasi). Jika solusi yang lebih baik telah diperoleh, prosedur simpleks dengan sendirinya akan menghilangkan semua solusi – solusi lain yang kurang baik.
c. Proses ini dilakukan berulang – ulang sampai suatu solusi yang lebih baik tak dapat ditemukan. Proses simpleks kemudian berhenti dan solusi optimum diperoleh.

Mengubah bentuk baku model pemrograman linear ke dalam bentuk tabel akan memudahkan proses perhitungan simpleks. Langkah – langkah perhitungan dalam algoritma simpleks adalah :

a. Berdasar bentuk baku, tentukan solusi awal, dengan menetapkan (n – m) variable nonbasis sama dengan nol. Dimana n jumlah variable dan m banyaknya kendala.
b. Pilih sebuah entering variable diantara yang sedang menjadi variabel nonbasis, yang jika dinaikkan diatas nol dapat memperbaiki nilai fungsi tujuan. Jika tak ada, berhenti berarti solusi sudah optimal. Jika tidak dilanjutkan ke langkah 1.
c. Plih sebuah leaving variabel diantara yang sedang menjadi variable basis yang harus menjadi nonbasis (nilainya menjadi nol) ketika entering variable menjadi variable basis.

d. Tentukan solusi yang baru dengan membuat entering variable dan leaving variable menjadi nonbasis. Kembali ke langkah 2.

2.5 Analisis dan Perancangan Sistem

Penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

Tahap analisis sistem dilakukan setelah tahap perencanaan sistem (*system planning*) dan sebelum tahap desain sistem (*system design*). Tahap analisis merupakan tahap yang kritis dan sangat penting, karena kesalahan di dalam tahap ini juga akan menyebabkan kesalahan di tahap selanjutnya.

Dalam tahap analisis sistem terdapat langkah-langkah dasar yang harus dilakukan oleh analis sistem sebagai berikut:

1. *Identify*, yaitu mengidentifikasi masalah.
2. *Understand*, yaitu memahami kerja dari sistem yang ada.
3. *Analyze*, yaitu menganalisis sistem.
4. *Report*, yaitu membuat laporan hasil analisis.

Setelah tahap analisis sistem selesai dilakukan, maka analis sistem telah mendapatkan gambaran dengan jelas apa yang harus dikerjakan. Tiba waktunya

sekarang bagi analis sistem untuk memikirkan bagaimana membentuk sistem tersebut. Tahap ini disebut dengan desain sistem.

Menurut Kendall & Kendall Brother (2003), Analisa dan Perancangan Sistem dipergunakan untuk menganalisis, merancang, dan mengimplementasikan peningkatan-peningkatan fungsi bisnis yang dapat dicapai melalui penggunaan sistem informasi terkomputerisasi.

A. Blok Masukan

Input mewakili data yang masuk ke dalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

B. Blok Model

Blok ini terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan keluaran yang diinginkan.

C. Blok Keluaran

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

D. Blok Teknologi

Teknologi merupakan “kotak alat” (*toolbox*) dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan.

E. Blok Basis Data

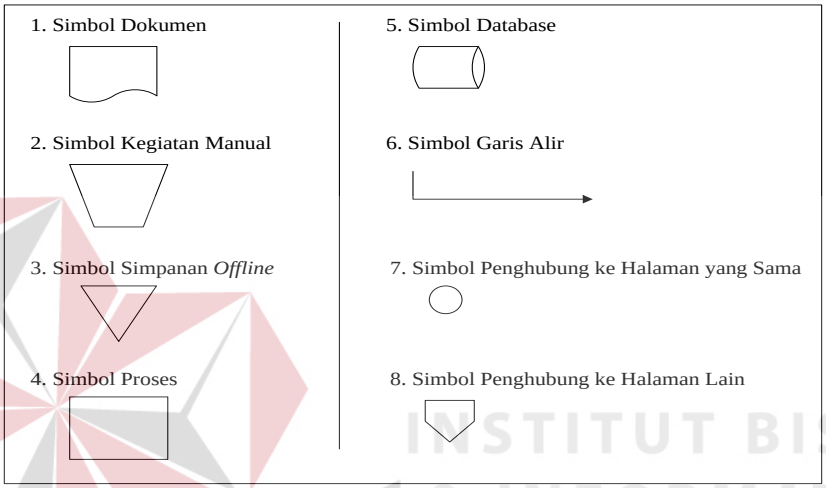
Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa, supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpannya. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut dengan DBMS (*Database Management Systems*).

F. Blok Kendali

Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidak-efisienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung diatasi.

2.5.1 *System Flow*

System flow atau bagan alir sistem merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. System flow menunjukkan urutan-urutan dari prosedur yang ada di dalam sistem dan menunjukkan apa yang dikerjakan sistem. Simbol-simbol yang digunakan dalam *system flow* ditunjukkan pada Gambar 2.1.



Gambar 2.1 Simbol-Simbol pada *System Flow.*

1. Simbol dokumen

   Menunjukkan dokumen input dan output baik untuk proses manual atau komputer.

2. Simbol kegiatan manual

   Menunjukkan pekerjaan manual.

3. Simbol simpanan *offline*

   Menunjukkan file non-komputer yang diarsip.

4. Simbol proses

Menunjukkan kegiatan proses dari operasi program komputer.

5. Simbol database

Menunjukkan tempat untuk menyimpan data hasil operasi komputer.

6. Simbol garis alir

Menunjukkan arus dari proses.

7. Simbol penghubung

Menunjukkan penghubung ke halaman yang masih sama atau ke halaman lain.

2.5.2 *Data Flow Diagram* (DFD)

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir. DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur dan dapat mengembangkan arus data di dalam sistem dengan terstruktur dan jelas.

A. *External Entity* atau *Boundary*

Kesatuan luar merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem. *External entity* disimbolkan dengan notasi kotak.

B. Arus Data

Arus Data (*data flow*) di DFD diberi simbol panah. Arus data ini mengalir di antara proses, simpanan data (*data store*) dan kesatuan luar (*external entity*). Arus data ini menunjukkan arus data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.

C. Proses

Suatu proses adalah kegiatan yang dilakukan oleh orang, mesin, atau komputer dari hasil suatu arus data yang masuk ke dalam proses untuk menghasilkan arus data yang akan keluar dari proses. Simbol proses berupa lingkaran atau persegi panjang dengan sudut-sudut tumpul.

D. Simpanan Data

Simpanan data merupakan simpanan dari data yang dapat berupa hal-hal sebagai berikut, sebagai gambaran:

1. Suatu file atau *database* di sistem komputer.
2. Suatu arsip atau catatan manual.
3. Suatu kotak tempat data di meja seseorang.
4. Suatu tabel acuan manual.

Simpanan data di DFD disimbolkan dengan sepasang garis horizontal paralel yang tertutup di salah satu ujungnya.

E. *Context Diagram*

Context Diagram merupakan langkah pertama dalam pembuatan *Data Flow Diagram*. Pada context diagram dijelaskan sistem apa yang dibuat dan entity apa saja yang digunakan. Dalam context diagram harus ada arus data yang masuk dan arus data yang keluar.

F. *Data Flow Diagram* (DFD) *Level* 0

DFD *level* 0 adalah langkah selanjutnya setelah *context diagram*. Pada langkah ini, digambarkan proses-proses yang terjadi pada Sistem Informasi Administrasi Keuangan Siswa.

G. *Data Flow Diagram* (DFD) *Level* 1

DFD *Level* 1 merupakan penjelasan dari DFD level 0. Pada proses ini dijelaskan proses apa saja yang dilakukan pada setiap proses yang terdapat di DFD level 0.

H. *Entity Relational Diagram* (ERD)

Entity Relational Diagram (ERD) merupakan penggambaran hubungan antara beberapa entity yang digunakan untuk merancang database yang akan diperlukan.

2.6. Siklus Hidup Pengembangan Sistem

Siklus Hidup Pengembangan Sistem 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).

2.6.1 Tahapan SDLC

1. Elisitasi Kebutuhan

Elisitasi atau pengumpulan kebutuhan merupakan aktivitas awal dalam proses reakayasa perangkat kebutuhan. Sebelum kebutuhan dapat dianalisis, dimodelkan, atau ditetapkan, kebutuhan harus dikumpulkan melalui proses elisitasi. Elisitasi kebutuhan adalah sekumpulan aktivitas yang ditujukan untuk menemukan kebutuhan suatu sistem melalui komunikasi dengan pelanggan, pengguna sistem dan pihak lain yang memiliki kepentingan dalam pengembangan sistem menurut Sommerville and Sawyer (1997).

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

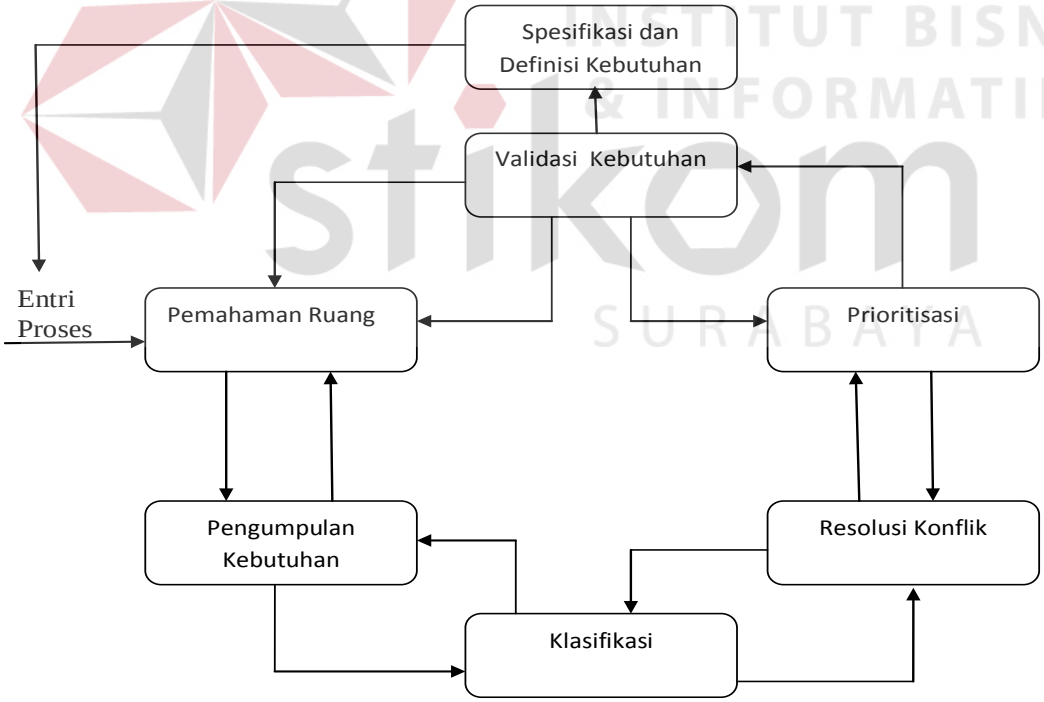
a) Mengetahui masalah apa saja yang perlu dipecahkan dan mengenali batasan-batasan sistem. Proses-proses dalam pengembangan perangkat lunak sangat ditentukan oleh seberapa dalam dan luas pengetahuan developer tentang permasalahan.

b) Mengenali siapa saja para *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 klien yang membayar sistem, pengembang yang merancang, membangun, dan merawat sistem, dan pengguna yang berinteraksi dengan sistem untuk mendapatkan hasil kerja mereka.

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* dari pada solusi yang dimungkinkan untuk masalah tersebut.

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

Pada Gambar 2.2, diagram tersebut menunjukkan tahapan-tahapan didalam analisis kebutuhan. Didalam proses rekayasa kebutuhan, analisis pun dilakukan dalam setiap aktivitas-aktivitasnya. Aktivitas tersebut antara lain:

a) *Domain Understanding*

Dalam tahapan ini, pengembang harus mengetahui bagaimana organisasi toko beroperasi dan apa yang menjadi permasalahan pada sistem yang berjalan.

b) *Requirements Collection*

Tahapan ini merupakan tahapan pengumpulan kebutuhan akan sistem yang akan dibangun sehingga diperlukan adanya interaksi secara intensif dengan *stakeholder*.

c) *Classification*

Tahapan ini mengelompokkan hasil dari tahap kebutuhan sehingga menjadi lebih terstruktur untuk selanjutnya diorganisir kedalam kelompok-kelompok yang koheren.

d) *Conflict Resolution*

Tahapan ini berguna untuk menemukan dan menyelesaikan kebutuhan yang didalamnya terdapat konflik. Konflik tersebut dapat terjadi antara dua *stakeholder* yang saling terkait tetapi memiliki fasilitas yang tidak sesuai, atau dapat terjadi antara kebutuhan dan sumber daya.

e) *Prioritisation*

Tahap ini melakukan interaksi dengan *stakeholder* untuk mengidentifikasikan kebutuhan-kebutuhan prioritas dari masing-masing kebutuhan agar memenuhi sumber daya yang tersedia pada organisasi.

f) *Requirements Checking*

Menganalisa sekumpulan kebutuhan dari hasil tahapan sebelumnya untuk menverifikasi dan memvalidasi berdasarkan aspek kelengkapan, konsistensi, dan kebutuhan nyata.

Semua jenis kebutuhan yang telah diperoleh tersebut kemudian dituangkan dalam bentuk dokumen yang berisi tentang kebutuhan sistem secara keseluruhan. Dokumen ini menjelaskan secara rinci tentang kesepakatan antara pengembang dengan klien, desain perangkat lunak yang akan dibangun, segala resiko yang akan dihadapi dan jadwal pembuatan perangkat lunak. Dokumen ini sangat berguna bagi pihak yang ingin mengetahui tentang perangkat lunak yang akan dibangun namun tidak mengerti secara teknik karena dokumen ini menggunakan bahasa yang sederhana. Secara umum dokumen ini biasa disebut dengan *Software Requirements Spesification* (SRS).

3. Desain

Tahap Desain adalah tahapan merancang pemodelan data yang dapat di visualisasikan melalui *Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM); dan pemodelan proses yang dapat di visualisasikan melalui *Data Flow Diagram* (DFD) atau melalui *Unified Modelling Language* (UML). Dalam tahap ini juga mentransformasikan hasil dari analisis kebutuhan menjadi kebutuhan yang sudah lengkap yang di fokuskan 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.

4. *Construction*

Tahap ini melakukan konversi hasil desain ke sistem informasi yang lengkap melalui tahapan *coding* atau pengkodean 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 pengetesan dan integritas.

Dalam proses pengimplementasian ini, digunakan beberapa aplikasi pendukung yaitu:

a) Microsoft Visual Basic .Net 2005

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. Didalam Visual Basic .NET 2005 ini, terdapat beberapa fasilitas yaitu antara lain fasilitas untuk penanganan kesalahan yang *real time background compiler* sehingga developer Visual C# dapat mengetahui kesalahan kode secara *up-to-date.*

b) SQL Server 2005

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk Microsoft. Bahasa kueri utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase. Umumnya SQL Server digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya SQL Server pada basis data besar.

5. Testing dan Implementasi

Tahap ini mendemonstrasikan sistem perangkat lunak yang telah selesai dibuat untuk dijalankan, apakah telah sesuai dengan kebutuhan yang telah di spesifikasikan 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 pengetesan dan objektif dari pengetesan. Pada tahap berikutnya adalah mendefinisikan *Test Techniques*, yaitu tentang bagaimana teknik yang digunakan termasuk dasar-dasar pengetesan berdasarkan intuisi dan pengalaman serta teknik pengetesan 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 di evaluasi kembali. Tahap terakhir adalah mendefinisikan *Test Process* yang berisi tentang aktivitas testing.

6. *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 di masa yang akan datang.

2.6.2 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.