BAB II

LANDASAN TEORI

2.1. Teori Tentang Permasalahan

2.1.1. Pengertian Simulasi

Simulasi adalah tiruan dari sebuah sistem dinamis dengan menggunakan model komputer yang digunakan untuk melakukan evaluasi dan meningkatkan kinerja sistem. Atau juga bisa disebut bahwasannya simulasi adalah proses yang diperlukan untuk operasionalisasi model, atau penanganan model untuk meniru tingkah laku sistem yang sesungguhnya.

Menurut Banks (1998:25) dalam bukunya simulasi adalah tiruan dari proses dunia nyata atau sistem, simulasi menyangkut pembangkitan proses serta pengamatan dari proses untuk menarik kesimpulan dari sistem yang diwakili. Selain itu Menurut Borow (1999:64) Simulasi adalah teknik untuk membuat konstruksi model suatu proses atau situasi, dalam rangka menduga secara karakteristik atau menyelesaikan masalah berkaitan dengannya dengan menggunakan model yang diajukan.

Jadi simulasi memprediksi atau mempelajari sesuatu yang belum terjadi dengan cara meniru atau membuat model sistem yang dipelajari dan selanjutnya mengadakan eksperimen dengan menggunakan komputer atau sejenisnya.

2.1.2. Sistem Dan Model Dalam Simulasi

2.1.2.1.Sistem

Sistem adalah sekumpulan objek yang tergabung dalam suatu interaksi dan inter-dependensi yang teratur. Dalam sistem itu sendiri terdapat beberapa komponen didalamnya yaitu:

1) Entitas : Objek yang diamati dari sebuah sistem
2) Atribut : Identitas dari Entitas
3) Aktivitas : Suatu masa yang mewakili proses suatu Entitas
4) Status : Kumpulan variabel yang dibutuhkan untuk menggambarkan sistem
5) Kejadian : Kejadian yang mengubah suatu sistem

2.1.2.2.Model

Model adalah suatu penyederhanaan dari sistem yang akan dipelajari. Model sangat beragam bisa dalam bentuk ikon, analog atau simbol. Model ikon meniru sistem nyata secara fisik, seperti *globe* (model dunia), *planetarium* (model sistem ruang angkasa). Model simbol tidak meniru sistem secara fisik atau tidak memodelkan perilaku sistem, tapi memodelkan sistem berdasarkan logikanya. Logika bisa bervariasi mulai dari intuisi ke bahasa verbal atau logika matematik. Karena model analisis simulasi harus dapat diimplementasikan pada komputer, maka model simulasi harus eksplisit, yaitu harus sebagai model simbolik paling tidak untuk level aliran logika.

Model simbolik dapat diklasifikasikan menjadi :

1) Model preskriptif atau deskriptif

   Model preskriptif digunakan untuk mendefinisikan dan mengoptimalkan permasalahan. Model deskriptif menggambarkan sistem berdasarkan perilakunya dan permasalahan optimasi diserahkan ke analisis berikutnya.

2) Model diskrit atau kontinu

   Pengklasifikasian model menjadi diskrit dan kontinu berdasarkan pada variabelnya. Perbedaan dalam kedua model adalah waktu. Jika revisi terhadap model terjadi secara kontinu berdasarkan waktu, maka model itu diklasifikasikan sebagai model kontinu.

3) Model probabilistik atau deterministik

   Perbedaan kedua model ini juga didasarkan pada variabel model. Jika ada variabel acak, maka kita klasifikasikan sebagai model probabilistik. Jika tidak, model merupakan klasifikasi model deterministik.

4) Model statis atau dinamis

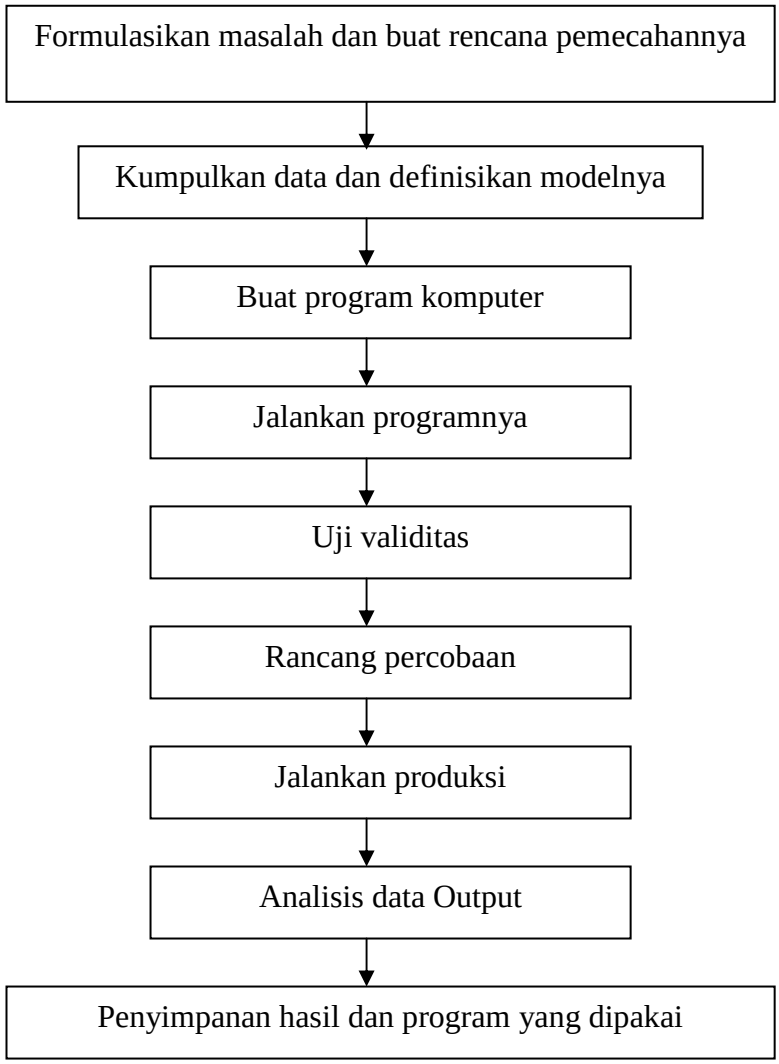
   Pembedaan kedua model ini didasarkan pada variabel model. Jika variabel model berubah sesuai dengan waktu, maka model digolongkan pada model dinamis.

5) Model loop terbuka atau tertutup

   Pengklasifikasian model kedalam bentuk loop terbuka atau tertutup didasarkan pada struktur model. Pada model terbuka output dari model

tidak menjadi umpan balik untuk memperbaiki input, sebaliknya adalah model loop tertutup.

Berikut adalah langkah-langkah dalam model simulasi :



2.1.3. Bahasa Simulasi

Pemrograman model simulasi, seperti yang disebutkan sebelumnya, dapat dilakukan menggunakan bahasa umum komputer (*general purposes language*) atau menggunakan bahasa simulasi. Pada bagian ini kita akan mempelajari beberapa bahasa simulasi, melihat dan memahami kelebihan dan kekurangan dari

masing-masingnya, sehingga kita melakukan pemilihan yang tepat saat kita perlu menggunakan bahasa simulasi.

Satu bahasa simulasi tidak dapat menjadi alat yang tepat untuk semua kasus permodelan simulasi. Pembahasan pada bagian ini akan kita mulai dengan mempelajari anatomi bahasa simulasi, lalu dilanjutkan dengan penjelasan berbagai bahasa simulasi lengkap dengan kelebihan dan kelemahannya.

2.1.3.1.Anatomi Bahasa Simulasi

Kesuksesan analisis simulasi merupakan teknik campuran yang sangat tergantung pada keahlian dan keahlian analis. Elemen dan struktur bahasa komputer umum seperti pascal atau fortran, sorce codenya tidak dengan mudah dapat digunakan untuk memodelkan simulasi sistem. Misalnya, bahasa itu tidak menyediakan struktur data yang enak digunakan untuk pemrosesan kejadian, sementara hal ini merupakan elemen logis yang sangat penting dalam permodelan simulasi. Tidak ada perintah dalam fortran misalnya yang dengan jelas menambah atau mengurangi antrian nasabah atau objek lainnya. Tidak ada perintah dalam fortran yang mengakumulasikan jumlah objek dalam antrian dan menghitung rata-rata untuk menyediakan output statistik penting.

Variabel waktu lanjut, yang penting dalam penjalanan model simulasi, juga tidak dapat ditemukan pada fortran dan bahasa pemrograman umum lainnya. Untuk memenuhi fungsi-fungsi di atas dan hal-hal penting lainnya dalam struktur model program komputer, kode pemrograman yang ekstensif, kompleks dan sulit didebug harus dibuat. Motivasi mengembangkan dan menggunakan bahasa simulasi berasal dari keinginan untuk mempersingkat waktu yang

dibutuhkan untuk mengembangkan model valid yang relatif mudah didebug dan yang menyediakan output statistik yang dibutuhkan dalam pengambilan keputusan. Bahasa simulasi pertama yang dihasilkan untuk tujuan itu adalah GPSS (*general purpose simulation system*) yang dikembangkan oleh Geoffrey Gordon dan dipublikasikan pertama sekali tahun 1961. bahasa ini telah berevolusi dalam beberapa versi, yang pada umumnya dikembangkan oleh IBM.

Pengembangan terpisah versi GPSS, GPSS/H memungkinkan debugging kode interaktif. Akhir-akhir ini, GPSS tersedia pada umumnya untuk mainframe dan minikomputer, dan ada 2 versi untuk mikrokomputer IBM. Elemen GPSS dikenal mempunyai derajat isomorfis tinggi dengan elemen sistem diskrit. GPSS diikuti dengan munculnya simscript tahun 1963, dikembangkan oleh perusahaan Rand. Bahasa ini memiliki kemampuan untuk permodelan sistem yang lebih kompleks. Untuk melakukan fungsi ini, elemen bahasa kurang jelas dihubungkan dengan dunia nyata. Penggunaan himpunan, kejadian, proses dan sumber daya menggambarkan secara utama pada struktur dan operasi program simscript. Bahasa-bahasa pionir ini tidak lama diikuti pengembangan bahasa-bahasa simulasi khusus lainnya dan jumlahnya sudah sangat banyak sampai saat ini.

2.1.3.2.Struktur Bahasa Simulasi

Kiviat mendefinisikan struktur statis bahasa simulasi terdiri dari 3 bagian, yaitu identifikasi objek, karakteristik objek, relasi antara objek dan penurunan objek. Struktur dinamisnya didefinisikan sebagai metode penambahan waktu simulasi. Objek adalah komponen model dan sistem yang menjadi perhatian

utama analisis, misalnya nasabah bank, komponen dalam lini perakitan, pengguna dalam sistem jaringan, dll. Bahasa yang berbeda memberikan definisi yang berbeda pada objek, misalnya dalam siman disebut entities, dalam GPSS disebut transactions. Masing-masing objek dalam sistem yang sama mempunyai karakteristik yang berbeda. Nasabah bank misalnya, ada yang ingin melakukan penarikan, ada yang ingin melakukan setoran, dll.

Pendefinisian karakteristik dalam bahasa yang berbeda juga berbeda. Karakteristik dalam siman dan simscript misalnya didefinisikan sebagai attributes sedangkan dalam GPSS didefinisikan sebagai parameters, dan ada juga yang menggunakan definisi properties, dll. Meskipun objek mempunyai karakteristik unik, untuk tujuan pemrosesan dalam model, ada baiknya karakteristik itu dikelompokkan. Setiap bahasa mempunyai mekanisme berbeda dalam melakukan pengelompokan ini. Bahkan dalam kasus sistem yang relatif kecil, mempertahankan semua objek dalam model selama penjalanan simulasi bisa tidak memungkinkan karena keterbatasan memori komputer.

Akibatnya, alat untuk menurunkan objek ketika dibutuhkan dan menghapusnya jika sudah tidak dibutuhkan harus disediakan. Cara setiap bahasa simulasi memfasilitasi ini sangat berbeda. Dalam beberapa kasus, mekanisme digunakan untuk menelusuri karakteristik akar bahasa kompiler darimana bahasa simulasi dikembangkan. Bahasa simulasi yang kurang dekat dengan konvensi struktur data dari kompiler tertentu menurunkan objek yang sangat mirip dengan sudut pandang dunia bahasa.

Sejalan dengan perbaikan kemampuan bahasa komputer umum (*general purposes*), bahasa simulasi khusus pada umumnya telah dikodean kembali seperti assembly, bahasa bebas mesin seperti C. Tetapi struktur awal penurunan objek tetap dalam bahasa simulasi. Struktur statis bahasa simulasi menempatkan objek dalam ruangan model, yaitu dimana objek secara fisik ditempatkan dalam sistem. Struktur dinamis dibutuhkan untuk menempatkan objek dalam waktu dan memungkinkan keberlanjutan dari satu titik waktu ke titik lainnya.

Seperti yang sudah dijelaskan dalam topik sebelumnya, ada dua pendekatan dasar yang digunakan dalam struktur dinamis, yaitu *fixed-time step* dan *event-tracking*. Pendekatan *fixed-time* memeriksa sistem pada interval waktu tetap untuk menentukan apakah statusnya sudah berubah atau belum. Jika status masih sama, variabel waktu akan ditambahkan sebesar interval waktu-tetap. Meskipun secara logika pendekatan ini cukup sederhana, tapi metodenya sangat tidak efisien. Mungkin ada beberapa titik waktu dimana sistem tidak berubah statusnya, dan karenanya akan ada banyak pemeriksaan sistem yang tidak perlu.

Akibatnya, tidak ada bahasa simulasi kejadian diskrit yang menggunakan pendekatan ini ke struktur dinamis. Pendekatan *event-tracking* memeriksa sistem hanya jika ada perubahan status. Logika dimasukkan dalam model untuk menentukan kapan kejadian atau status sistem berubah, dan variabel waktu ditambahkan dengan tepat sampai titik sebelum sistem diperiksa. Logika yang dibutuhkan untuk melakukan ini lebih kompleks dibandingkan dengan langkah waktu-tetap, tetapi akan menghemat waktu eksekusi model secara signifikan.

2.1.3.3.Karakteristik Bahasa Simulasi

Struktur dinamis dan statis bahasa simulasi menyediakan kebutuhan jelas untuk mengeksekusi model simulasi. Beberapa sifat bahasa simulasi menurut Banks (1998:25) :

1) Pengembangan kode model: Kebanyakan bahasa simulasi masih membutuhkan pemasukan pernyataan kode untuk menciptakan kode model, tetapi kemampuan grafik mikrokomputer telah memungkinkan input grafik. Cara ini paling sesuai untuk bahasa yang fokus pada aliran objek melalui elemen atau blok model.
2) *Debugging* model: Begitu mode simulasi sudah dikodekan menggunakan bahasa simulasi yang dipilih, langkah selanjutnya adalah debugging kode sehingga model simulasi berjalan ke penghentian normal. *Syntax errors* (kesalahan sintaks) adalah permasalahan pertama dalam proses, dan analisis untuk mendeteksi ini sudah ditanam dalam bahasa simulasi umumnya. Kesulitan berikutnya yang dihadapi adalah perbaikan kesalahan selama eksekusi kode. Analisis bahasa simulasi umumnya tidak sesuai secara total dengan permasalahan ini. Setelah menemukan kesalahan seperti ini, program berhenti dan tidak memberikan alasan dalam bentuk logika model kenapa program berhenti.
3) Penurunan variabel acak: Untuk kebanyakan simulasi probabilistik, kemampuan mengekstrak sampel acak dari distribusi probabilitas

tertentu sangat penting. Bahasa simulasi melakukannya dengan mudah.

4) Pengumpulan statistik: Penjalanan model simulasi tanpa mengumpulkan data ukuran kinerja sistem sama saja dengan tidak melakukan pengamatan pada sistem dunia nyata yang sedang berlangsung. Pengamat ada selama operasi sistem dunia nyata tetapi tidak mengamati dan mencatat apa yang terjadi. Bahasa simulasi harus memungkinkan pengguna dengan mudah menspesifikasikan beragam statistik yang dikumpulkan selama eksekusi model. Juga untuk membantu interpretasi output simulasi kemampuan penggambaran grafik dan inferensi statistik diperlukan.
5) Disain percobaan: Karena analisis simulasi bersifat deskriptif, kesuksesan aplikasinya tergantung pada percobaan model. Rancangan percobaan efektif dan efisien benar-benar meningkatkan kualitas solusi yang didapatkan dari model simulasi.
6) Animasi grafis dan output dinamis: Kemampuan menggunakan bahasa simulasi pada mikrokomputer memungkinkan kemampuan grafis mesin ini untuk mengilustrasikan penjalanan mode simulasi atau outputnya. Ilustrasi objek yang mengalir melalui elemen model disebut sebagai animasi. Animasi biasanya menggunakan monitor berwarna dan dengan mudah mengenali simbol objek dan elemen model. Dengan mengamati aliran seperti itu, analisis dapat memperhatikan penyebab permasalahan operasi dan dapat

memperbaikinya. Animasi model akan memperlambat eksekusi model. Oleh karena itu, animasi biasanya hanya dilakukan pada mikrokomputer cepat dengan memori besar.

2.1.3.4.Elemen Simulasi

Analisis simulasi merupakan teknik pemodelan deskriptif, karena itu tidak ada formulasi permasalahan dan langkah penyelesaian eksplisit yang merupakan bagian integral dari model optimasi.

Meskipun tidak ada langkah eksplisit, paling tidak kita dapat menggunakan elemen simulasi berikut dalam perancangan model simulasi:

1) **Formulasi permasalahan**

Formulasi Masalah merupakan suatu langkah yang sangat penting dalam perancangan model simulasi. Formulasi masalah yang tidak tepat tidak akan mungkin menghasilkan model yang tepat (akurat). Formulasi masalah merupakan suatu kegiatan untuk memilih satu permasalahan yang dianggap paling penting untuk diselesaikan saat itu dari sekian banyak permasalahan.

Hal-hal berikut diungkapkan dalam formulasi masalah:

a. Identifikasi keputusan dan variabel tidak dapat dikontrol
b. Spesifikasi pembatas variabel keputusan
c. Mendefinisikan ukuran kinerja sistem dan fungsi tujuan
d. Mengembangkan model struktur awal yang menghubungkan variabel sistem dan ukuran kinerja

2) Variabel dan pembatas

Setelah manajemen memutuskan permasalahan yang akan diselesaikan dalam model simulasi, maka langkah selanjutnya adalah menentukan variable yang mendefinisikan sistem dan outputnya. Variabel dapat dikategorikan sebagai variabel eksogenus dan endogenus. Variabel eksogenus kadang-kadang disebut juga sebagai variabel input sedangkan variabel endogenus disebut juga sebagai variabel output.

a. Variabel eksogenus : ada di luar sistem dan tidak terikat dengan model.

b. Variabel endogenus : ada dalam sistem dan merupakan fungsi variable eksogenus.

Variabel eksogenus terdiri dari variabel yang dapat dikontrol dan tidak dapat dikontrol. Variabel eksogenus yang dapat dikontrol dapat dimanipulasi pengambil keputusan, sedangkan variabel eksogenus yang tidak dapat dikontrol tidak dapat dimanipulasi pengambil keputusan.

Jadi untuk menyelesaikan permasalahan yang dihadapi manajemen, menggunakan model simulasi, mereka hanya dapat memanipulasi variabel eksogenus yang dapat dikontrol. Penentuan variabel sebagai terkontrol atau tidak tergantung dari kemampuan pengambil keputusan mengendalikan sumber daya.

Variabel eksogenus yang dapat dikontrol kadang-kadang disebut dengan variabel keputusan. Variabel eksogenus yang tidak dapat dikontrol kadang-kadang disebut dengan parameter sistem. Sumber daya yang membatasi dalam mencapai

tujuan juga harus didefinisikan dengan tepat. Permasalahan timbul karena adanya batasan-batasan dalam sistem.

3) Pengukuran kinerja sistem dan fungsi tujuan

Ukuran kinerja sistem bisa lebih dari satu. Pengoptimalan salah satunya bisa saling bertentangan dengan ukuran kinerja lainnya. Pengambil keputusan harus dapat memilih ukuran kinerja yang paling tepat untuk tujuan optimasi.

4) Pengumpulan Data

Data diperlukan untuk percobaan model. Verifikasi dan validasi model dapat dilakukan dengan adanya data. Dalam validasi dan verifikasi, analisis menguji seberapa dekat model yang dibuat dapat meniru sistem aslinya dengan membandingkan output model dengan kinerja sistem.

Output akan diperoleh jika simulasi dijalankan untuk data tertentu. Data bisa diperoleh dengan pengamatan dan pelaporan pribadi, atau dengan membangkitkan bilangan acak jika data historisnya sudah ada. Cara kedua ini khususnya digunakan untuk model probabilistik. Ukuran sampel tergantung dari biaya yang bersedia dikeluarkan untuk keakuratan tertentu.

5) Pengembangan Model

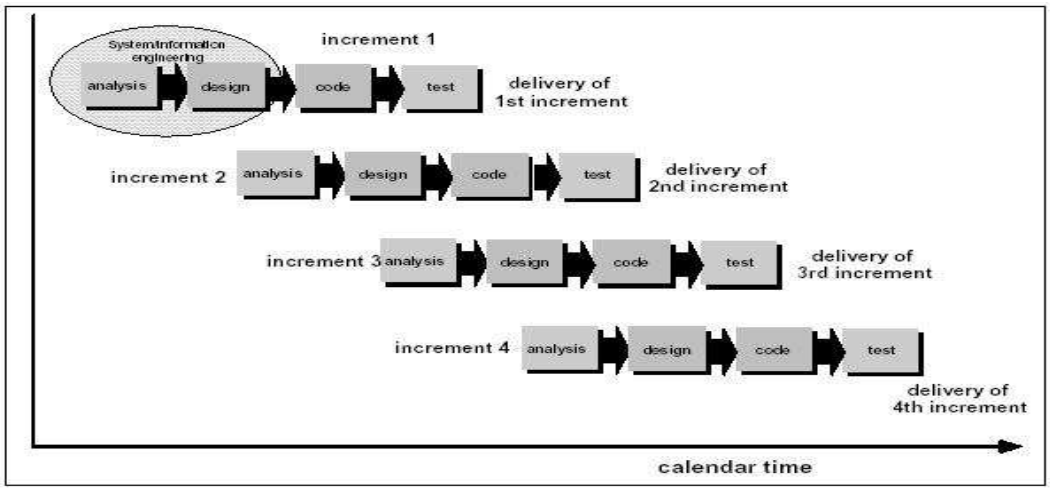
Pemahaman yang baik akan sistem sebenarnya sangat diperlukan dalam membentuk model dan merupakan hal yang sulit juga untuk dilakukan. Tidak ada pendekatan yang baku dalam membentuk model. Ada dua pendekatan yang dapat kita gunakan, yaitu pendekatan aliran fisik dan perubahan status. Dalam pendekatan aliran fisik, pemrosesan atau perpindahan entiti secara fisik ditunjukkan dalam model.

Keberadaan *entity* ini dilacak dalam sistem selama proses penjalanan simulasi, untuk mengetahui entiti sedang diproses dimana dan pencabangan aturan keputusan untuk menentukan rutenya. Diagram alur entiti dan pemrosesan elemen sistem memberikan representasi sistem darimana model dan pemrograman komputernya dikembangkan. Dalam pendekatan perubahan status, kita memerlukan variabel status (termasuk dalam klasifikasi variabel endogenus) dan kejadian. Kita dapat bedakan model yang akan kita bangun ke dalam model konseptual, logika dan simulasi. Penggolongan ini akan memudahkan dalam membentuk model simulasinya :

a. Model konseptual : Menggambarkan sistem secara konsep, dapat secara verbal atau menggunakan grafik.
b. Model logika : Menerjemahkan model konseptual ke dalam bentuk suatu diagram alur atau algoritma.
c. Model simulasi : Menerjemahkan model logika ke dalam program komputer

2.2. Metode Pengembangan Sistem

Berkaitan dengan langkah pembuatan Perangkat Lunak, penulis menggunakan model proses *increment models* untuk membantu dalam proses pengembangan sistem dan perangkat lunak dan juga sebagai landasan untuk penelitian yang akan dilakukan. Berikut dijelaskan mengenai metode pengembangan sistem yang penulis pakai (*increment models*) :



Gambar 2.1 *Increment Models*

[Sumber : Sommerville, ian. " Software engineering" 6th. Addison weyley. 2001]

Keterangan gambar diatas menurut Somer (2001:17) :

1) Kombinasikan elemet-element dari *incremental* dengan sifat iterasi/perulangan.
2) Element-element dalam *incremental* dikerjakan dengan hasil berupa produk dengan spesifikasi tertentu, kemudian proses dimulai dari fase pertama hingga akhir dan menghasilkan produk dengan spesifikasi yang lebih lengkap dari yang sebelumnya. Demikian seterusnya hingga semua spesifikasi memenuhi kebutuhan yang ditetapkan oleh pengguna.
3) Produk hasil *increment* pertama biasanya produk inti (*core product*), yaitu produk yang memenuhi kebutuhan dasar. Produk tersebut digunakan oleh pengguna atau menjalani *review* detil. Hasil *review* tersebut menjadi bekal untuk pembangunan pada *increment*

berikutnya. Hal ini terus dikerjakan sampai produk yang komplit dihasilkan.

4) Model ini cocok jika jumlah anggota tim pengembang / pembangun perangkat lunak tidak cukup.
5) Mampu mengakomodasi perubahan secara fleksibel.
6) Produk yang dihasilkan pada *increment* pertama bukanlah *prototype*, tapi produk yang sudah bisa berfungsi dengan spesifikasi dasar.

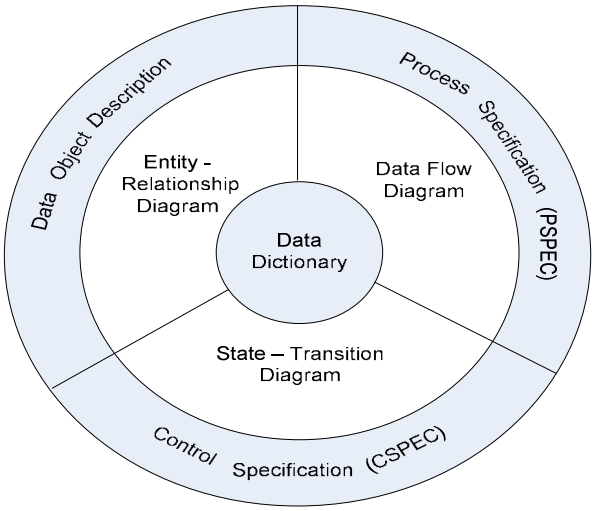
2.3. Analisis dan Perancangan Terstruktur

Di dalam analisis dan perancangan terstruktur akan dijelaskan mengenai model analisis maupun alat bantu dalam analisis dan perancangan. Berikut akan dijelaskan keduanya :

2.3.1. Elemen Model Analisis

Menurut Rog (1997:07) Model analisis harus dapat mencapai tiga sasaran utama yaitu :

1) Untuk menggambarkan apa yang dibutuhkan pelanggan
2) Untuk membangun dasar bagi pembuatan desain perangkat lunak
3) Untuk membatasi serangkaian persyaratan yang dapat divalidasi begitu perangkat lunak dibangun. Untuk mencapai sasaran tersebut, model analisis yang ditarik selama analisis terstruktur berlangsung ditunjukkan pada gambar berikut :



Gambar 2.2 Struktur model analisis

[Sumber : Roger S. Pressman, 1997]

2.3.2. Alat Bantu Metode Analisis dan Perancangan Terstruktur

Di dalam pendekatan sistem, penulis menggunakan alat bantu dalam metode analisis dan perancangan terstruktur, yang menghendaki adanya gambaran terhadap keseluruhan sistem menggunakan alat bantu seperti Flowchart.

2.3.2.1.Flowchart

Flowchart adalah teknik penyusunan instruksi untuk penulisan program komputer terstruktur dengan menggunakan gambar-gambar / simbol-simbol. Tujuan utama dari penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi dan jelas dengan menggunakan simbol-simbol standart.

Adapaun karakteristik ynag dimiliki *flowchart* antara lain :

1) *Flowchart* bersifat grafis atau menggunakan gambar-gambar sebagai simbol / lambang untuk suatu jenis pekerjaan / tugas / fungsi tertentu.

2) Perintah bersifat esensial, yaitu hanya perintah yang penting-penting saja yang digambarkan dalam *flowchart*.
3) *Flowchart* efektif untuk merancang program terstruktur (*structured programs*).

Pedoman yang perlu diperhatikan dalam membuat *flowchart* adalah:

1) Bagan alir sebaiknya digambar dari atas ke bawah dan mulai dari bagian kiri dari suatu halaman.
2) Kegiatan di dalam bagan alir harus ditunjukkan dengan jelas.
3) Harus ditunjukkan dari mana kegiatan akan dimulai dan dimana akan berakhir.
4) Masing-masing kegiatan di dalam bagan alir sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan.
5) Masing-masing kegiatan di dalam bagan lair harus didalam urutan yang semstinya.
6) Kegiatan yang terpotong dan akan disambung ke tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.
7) Gunakan simbol-simbol alir yang standart.

2.4. Basis Data

Pengertian Basis Data menurut Jog (2001:51), yaitu “Basis data (*database*) adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam simpanan luar computer, dan menggunakan perangkat lunak untuk memanipulasinya”. Proses dasar yang dimiliki oleh database ada 4, yaitu :

1) Pembuatan data-data baru (*create database*).
2) Penambahan data (*insert*)
3) Mengubah data (*update*)
4) Menghapus data (*delete*)

Pemanfaatan basis data untuk memenuhi sejumlah tujuan / objektif sebagai berikut :

1) Kecepatan dan kemudahan (*Speed*)
2) Efisiensi Ruang Penyimpanan (*Space*)
3) Keakuratan (*Accurancy*)
4) Ketersediaan (*Availability*)
5) Kelengkapan (*Completeness*)
6) Keamanan (*Security*)
7) Kebersamaan Pemakaian (*Shatability*)

Adapun permasalahan yang sering timbul dalam penyusunan basis data adalah sebagai berikut :

1) Data Redudansi, yaitu munculnya data-data yang sama secara berulang-ulang pada file basis data yang semestinya tidak diperlukan.
2) Data Tidak Konsisten, yaitu munculnya data yang tidak konsisten pada atribut yang sama untuk beberapa file yang kuncinya sama.
3) Data Terisolasi, disebabkan oleh pemakaian beberapa file basis data, dimana program aplikasi tidak dapat mengakses data-data dari file tertentu.

4) Keamanan Data, bahwa data-data dalam basis data merupakan sumber informasi yang bersifat sangat penting dan rahasia.
5) Kesatuan Data, dimaksudkan sebagai suatu sarana untuk meyakinkan bahwa data-data yang tersimpan dalam basis data selalu berada dalam kondisi yang benar, *up to date*, konsisten dan selalu tersedia.

2.5. Pengujian Sistem

Pengujian menyajikan anomali yang menarik bagi perekayasa perangkat lunak. Pada proses perangkat lunak, perekayasa pertama–tama berusaha membangun perangkat lunak dari konsep abstrak ke implementasi yang dapat dilihat, baru kemudian dilakukan pengujian.

2.5.1. Sasaran-Sasaran Pengujian

Terdapat sejumlah aturan yang berfungsi sebagai sasaran pengujian :

1) Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2) Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3) Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

2.5.2. Prinsip Pengujian

Menurut Moek (1994:25) Sebelum mengaplikasikan metode untuk mendesain test case yang efektif, perekayasa perangkat lunak harus memahami

prisnsip dasar yang menuntun pengujian perangkat lunak. Serangkaian prinsip pengujian akan dijelaskan berikut ini :

1) Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan
2) Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
3) Prinsip pareto berlaku untuk pengujian perangkat lunak.
4) Pengujian harus mulai dari yang kecil dan berkembang ke pengujian yang besar
5) Pengujian yang mendalam tidak mungkin.
6) Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang independent.

2.5.3. Pengujian Black – Box

Menurut Rog (1997:44) Pengujian black-box berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian black-box memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program.

Pengujian black-box berusaha menemukan kesalahan dalam kategori sebagai berikut :

1) Fungsi-fungsi yang tidak benar atau hilang,
2) Kesalahan interface
3) Kesalahan dalam struktur data atau akses eksternal
4) Kesalahan kinerja

5) Inisialisasi dan kesalahan terminasi

2.5.4. Pengujian Unit

Menurut Rog (1997:65) Strategi pengujian perangkat lunak dimulai dengan *unit testing*, *integration testing*, *validation testing*, dan *sistem testing.*

Salah satu strategi pengujian perangkat lunak adalah pengujian unit, yaitu berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak yang disebut modul. Modul diuji untuk memastikan bahwa informasi secara tepat mengalir masuk dan keluar dari inti program yang diuji.

Pengujian modul didesain untuk mengungkap kesalahan sehubungan dengan komputasi yang salah. Kesalahan umum dalam komputasi adalah:

1) Kesalahpahaman atau preseden aritmatik yang tidak benar
2) Operasi mode yang tercampur
3) Inisialisasi yang tidak benar
4) Akurasi ketelitian
5) Representasi simbolis yang tidak benar dari sebuah persamaan

Dalam pengujian unit juga harus mengungkap kesalahan kesalahan-kesalahan yang terjadi seperti:

1) Perbandingan tipe data yang berbeda
2) Preseden atau operator logika yang tidak benar
3) Pengharapan akan persamaan bila precision error membuat persamaan yang tidak mungkin
4) Perbandingan atau variabel yang tidak benar
5) Penghentian loop yang tidak ada atau tidak teratur

6) Kegagalan untuk keluar saat terjadi iterasi divergen

7) Variabel loop yang dimodifikasi secara tidak teratur

Pengujian unit ini berkonsentrasi pada verifikasi fungsional dari sebuah modul dan gabungan modul-modul ke dalam struktur program. Setelah melakukan pengujian unit diharapkan secara modul tidak menemukan lagi kesalahan.

2.6. Sekilas tentang Visual Basic

Menurut Fir (2001:05) dalam bukunya yang berjudul 7 jam Belajar *Visual Basic* Untuk Orang Awam : *Visual Basic* adalah sebuah bahasa pemograman yang sangat mudah untuk dimengerti dan sangat popular, *Visual Basic* diciptakan pada tahun 1991 oleh *Microsoft* untuk menggantikan bahasa pemograman *Basic.*

Visual Basic diperkenalkan pada 1991. Pendekatan bagi menghubungkan bahasa pengaturacaraan kepada antaramuka pengguna grafik diambil dari prototaip yang dimajukan oleh Alan Cooper yang dikenali sebagai *Tripod.*

Microsoft menggajikan Cooper dan rakan niaganya untuk memajukan Tripod kepada kerangka boleh program bagi Windows 3.0, di bawah nama kod *Ruby* (tiada kaitan dengan bahasa pengaturcaraan Ruby).

Tripod tidak disertakan dengan bahasa pengaturcaraan sama sekali, dan Ruby mengandungi hanya penproses perintah asas mencukupi bagi peranannya sebagai kerangka Windows. Microsoft memutuskan untuk menggunakan kerangka program manager mudah bagi Windows 3.0 bukannya Ruby, dan mengabungkan Ruby dengan bahasa Basic untuk menghasilkan Visual Basic.

2.7. Database Microsoft Access

Microsoft Access (*Microsoft Office Access*) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi *Microsoft Office*, selain tentunya *Microsoft Word*, *Microsoft Excel*, dan *Microsoft PowerPoint*. *Microsoft Access* menggunakan mesin basis data *Microsoft Jet Database Engine*, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. Versi terakhir adalah *Microsoft Office Access 2007* yang termasuk ke dalam *Microsoft Office System 2007*.

Microsoft Access dapat menggunakan data yang disimpan di dalam format *Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database*, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna / programmer yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. Access juga mendukung teknik-teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.