BAB II

LANDASAN TEORI

2.1 Pengertian Sistem Aplikasi

Menurut Jogianto (2004), sistem berasal dari bahasa latin " Systema" dan bahasa Yunani "Sustema" yang berarti "satu kesatuan yang atas komponen atau elemen-elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi". Sistem adalah beberapa komponen yang saling berhubungan, bekerja sama untuk mencapai tujuan dengan menerima *input* dan menghasilkan *output*. Dari definisi sistem diatas, dapat disimpulkan bahwa sistem adalah suatu jaringan yang saling berhubungan dan saling memiliki keterkaitan antara bagian dan prosedur-prosedur yang ada terkumpul dalam satu organisasi untuk melakukan kegiatan untuk mencapai suatu tujuan tertentu.

2.2 Analisis sistem

Menurut Yakub (2012), Analisa sistem suatu proses yang dilakukan untuk menganalisis jabatan dan tugasnya, proses bisnis perubahan, ketentuan dan aturan perusahaan, masalah yang dihadapi perusahaan dan mencari solusinya serta rencana-rencana yang diinginkan oleh perusahaan.

Menurut Mulyato (2009), Analisa sistem adalah teori yang digunakan untuk landasan konseptual yang bertujuan untuk memperbaiki fungsi-fungsi yang ada di dalam sistem yang masih berjalan agar sistem tersebut menjadi lebih baik, lebih efisien dan mencapai tujuan yang harapan dengan cara mengubah sasaran

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Sistem yang sedang berjalan, mengganti *output* yang sedang digunakan, dan lain-lainnya.

2.3 Tahap-tahap Analisis Sistem

Menurut Mulyanto (2009), tahapan analisis sistem merupakan tahapan yang sangat penting karena tahapan ini dapat menyebabkan kesalahan pada tahap berikutnya apabila ditahapan ini terjadi kesalahan. Tahapan ini merupakan tahapan yang sangat mudah apabila *client* sangat paham dengan permasalahan yang dihadapi oleh organisasinya dan tahu betul fungsi-fungsi dari sistem informasi yang akan dibuat. Tetapi tahap ini juga tahap yang sangat sulit apabila *client* tidak mengetahui permasalahan yang dihadapi atau *client* tidak dapat mengidentifikasi permasalahan dan tertutup terhadap pihak luar yang ingin mengetahui detail-detail proses bisnisnya.

Menurut Mulyanto (2009), Di dalam tahap analisis sistem terdapat langkah-langkah yang harus dilakukan oleh seorang analis sistem, diantaranya adalah:

1. *Identify*, yaitu proses yang dilakukan untuk mengidentifikasi masalah.
2. *Understand* yaitu memahami kerja dari sistem yang ada.
3. *Analysis*, yaitu melakukan analisis terhadap sistem.
4. *Report*, yaitu membuat laporan dari hasil analisis yang telah dilakukan dalam kurun waktu tertentu.

2.4 Pengertian Penjadwalan

Menurut Sam'ani (2012), penjadwalan adalah penempatan sumber daya (*resource*) dalam satu waktu. Penjadwalan mata kuliah bertujuan untuk

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

menjadwalkan pertemuan dari beberapa sumber daya. Sumber daya yang dimaksud yaitu dosen yang mengajar mata kuliah, mata kuliah, ruang kuliah, kelas mahasiswa dan waktu perkuliahan. Di dalam penjadwalan mata kuliah terdapat batasan/persyaratan dalam penyusunannya yaitu dosen tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan dan satu ruang dan ruang tidak dijadwalkan lebih dari satu kali pada waktu bersamaan. Jika terjadi pelanggaran terhadap kendala yang ditetapkan maka akan mendapatkan nilai penalti untuk setiap pelanggaran. Semakin kecil jumlah pelanggaran yang terjadi solusi penjadwalan yang dihasilkan akan semakin baik.

Menurut Arviani (2013), kebanyakan orang terbiasa dengan jadwal sekolah yang disajikan sebagai tabel hari dalam seminggu dan slot waktu. Dapat dilihat bahwa setiap hari dibagi ke dalam slot waktu. Setiap slot waktu memiliki daftar mata pelajaran yang sedang diajarkan, oleh siapa dan di mana. Jadwal dapat dinyatakan dalam sejumlah cara yang berbeda, masing-masing siswa harus memiliki jadwal sendiri tergantung pada mata pelajaran, begitu juga masing-masing guru dan ruang, semua ini adalah perspektif yang berbeda pada jadwal yang sama. Lebih jelasnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Jadwal Mata Pelajaran (Alviani, 2013)

Hari	09.00 - 10.00	10.00 - 11.00	11.00 - 12 .00
Senin	Matematika Smith Ruang 1 Kelas A	Komputer Carlie Ruang 1 Kelas C	Biologi Donald Ruang 2 Kelas A
Selasa	Kimia Donald Ruang 1 Kelas A	Fisika Donald Ruang 3 Kelas A	Geografi Carlie Ruang 2 Kelas A

2.5 Pengertian Penjadwalan Mata Kuliah

Menurut Arviani (2013), Penjadwalan mata kuliah adalah kegiatan administratif yang paling utama di universitas. Dalam masalah penjadwalan mata kuliah, sejumlah mata kuliah yang dialokasikan ke sejumlah ruang yang tersedia dan sejumlah slot waktu disertai dengan constraints. *Constraints* terbagi atas dua jenis, yaitu *hard constraints* dan *soft constraints*

Hard constraints merupakan batas-batas yang harus diterapkan pada penjadwalan mata kuliah dan harus dipenuhi. Solusi yang tidak melanggar *hard constraints* disebut solusi layak. *Hard constraints* yang umum dalam penjadwalan mata kuliah adalah sebagai berikut:

a. Seorang dosen hanya dapat memberi kuliah untuk satu lokasi pada waktu tertentu.
b. Seorang mahasiswa hanya dapat mengikuti kuliah untuk satu lokasi pada waktu tertentu.
c. Sebuah lokasi (ruangan) hanya dapat digunakan untuk satu mata kuliah pada waktu tertentu.
d. Mahasiswa tidak dapat dialokasikan pada suatu lokasi yang menyebabkan lokasi melebihi kapasitas maksimum.

Soft constraints didefinisikan sebagai batas-batas mengenai alokasi sumber daya yang jika dilanggar masih dapat menghasilkan solusi yang layak. Dalam kenyataannya, masalah penjadwalan mata kuliah biasanya tidak mungkin untuk memenuhi semua *soft constraints*. Kualitas jadwal yang layak dapat dinilai berdasarkan seberapa baik *soft constraints* dapat dipenuhi. Namun, beberapa

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

masalah yang kompleks sulit menemukan solusi yang layak. Sebagai contoh, *soft constraints* yang mungkin ingin dicapai dalam jadwal sehubungan dengan aspek mata kuliah adalah meminimalkan terjadinya jadwal mata kuliah satu tingkat yang beturut-turut.

Menurut Arviani (2013), Beberapa universitas dengan jumlah mata kuliah yang akan dijadwalkan dan berbagai *constraints* yang harus dipertimbangkan membuat penyusunan jadwal mata kuliah menjadi sangat sulit.

2.6 Batasan-Batasan Dalam Masalah Penjadwalan Mata Kuliah

Menurut Arviani (2013), Dalam masalah penjadwalan memiliki beberapa macam batasan yang dapat menyebabkan *output* yang dihasilkan menjadi salah. Dalam menerapkan batasan dalam suatu masalah, biasanya tidak terlalu sama untuk setiap masalah. Batasan tersebut terdiri dari:

1. *Edge constraint*

   Edge constraint adalah batasan yang mengatur dua kejadian tidak boleh menempati satu slot waktu yang sama. Contohnya pada hari Senin jam 07.30 sampai 08.10 tidak mungkin dosen A mengajar di ruang 1-A dan mengajar di ruang 5-A.

2. *Present specification and exclusion*

   Present specification and exclusion adalah menentukan terlebih dahulu slot waktu yang akan digunakan oleh suatu kejadian sebelum proses pencarian solusi dilakukan. Contoh mata pelajaran agama digabung dengan kelas 1-A dan 1-B maka akan ditentukan waktu yang sama untuk kedua kelas tersebut agar tidak terganggu untuk menyusun mata pelajaran yang lain.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

3. *Capacity constraint*

Capacity constraint adalah batasan yang berhubungan dengan kapasitas ruangan. Untuk masing-masing kelas hanya boleh diisi sebanyak 40 siswa.

4. *Hard and soft constraint*

Hard constraint adalah batasan yang sama sekali tidak boleh dilanggar, sedangkan *soft constraint* adalah batasan yang diusahakan semaksimal mungkin tidak dilanggar namun jika dilanggar, hal tersebut masih dapat diterima.

2.7 Metode Algoritma Genetika

Menurut Sam'ani (2012), Algoritma Genetika adalah sebuah algoritma yang berbasis tentang mekanisme seleksi alam dan genetika. Algoritma Genetika ini menggunakan teori-teori dalam ilmu biologi, sehingga di dalam Algoritma Genetika terdapat istilah-istilah dan kosep biologi yang digunakan dalam Algoritma Genetika. Karena sesuai dengan namanya, proses-proses yang terjadi yang terjadi di dalam algoritma sama dengan yang terjadi pada evaluasi biologi. Di dalam Algoritma Genetika terdapat beberapa struktur umum yaitu solusi, kromosom, pindah silang, mutasi, *elitisme*, kondisi seleksi. Lebih jelas dapat dilihat pada Gambar 1 sebagai berikut:

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Ada 3 keuntungan utama dalam mengaplikasikan Algoritma Genetika pada masalah-masalah optimasi (Sam'ani, 2012):

a. Algoritma Genetika tidak memerlukan kebutuhan matematis banyak mengenai masalah optimasi.
b. Kemudahan dan kenyamanan pada operator-operator evolusi membuat Algoritma Genetika sangat efektif dalam melakukan pencarian global.
c. Algoritma Genetika menyediakan banyak fleksibel untuk digabungkan dengan metode *heuristic* yang tergantung domain, untuk membuat implementasi yang efisien pada masalah-masalah khusus.

2.8 Istilah dalam Algoritma

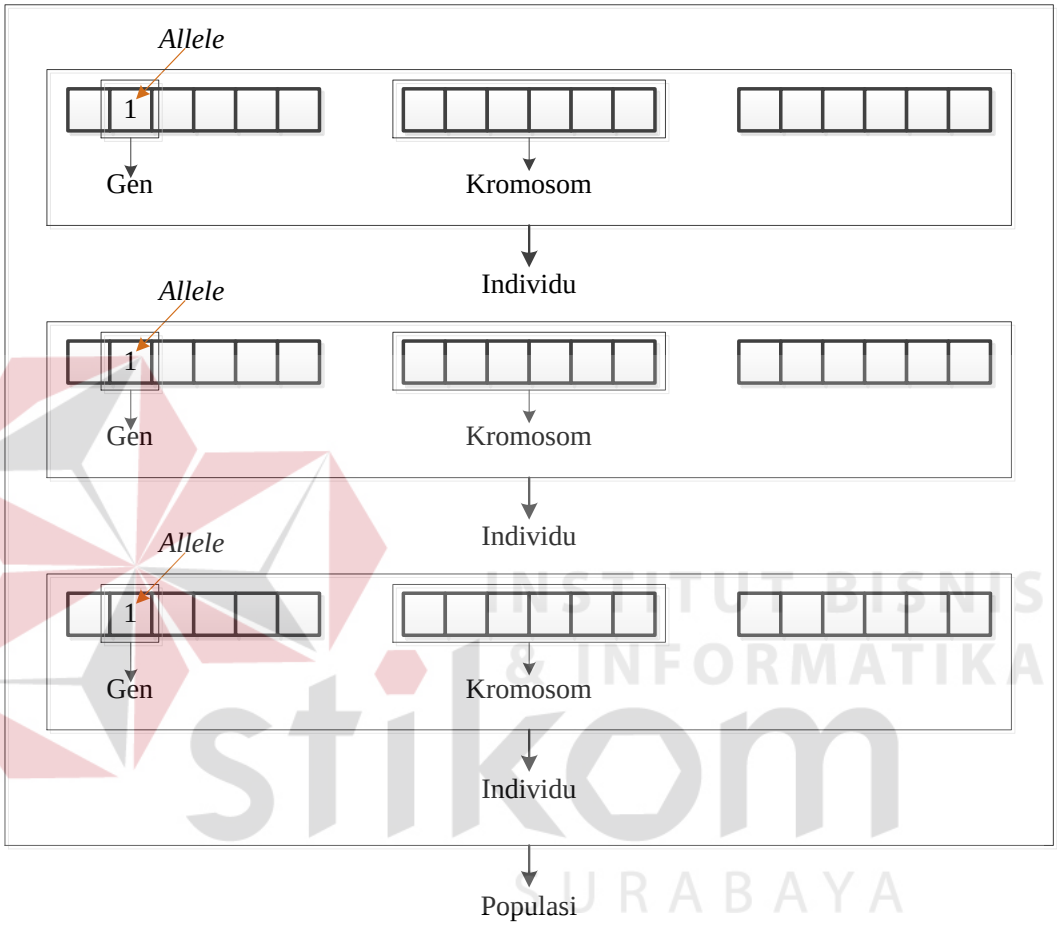
Menurut Sam'ani (2012), terdapat beberapa definisi penting dalam Algoritma Genetika yang perlu diperhatikan, yaitu:

a. *Genotype* (Gen), sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa berupa *biner*, *float*, *integer* maupun karakter, atau kombinatorial
b. *Allele*, merupakan nilai dari gen
c. Individu atau kromosom, gabungan gen-gen yang membentuk nilai tertentu dan merupakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
d. Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evaluasi.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

e. Generasi, menyatakan satu siklus proses evolusi atau satu iterasi di dalam Algoritma Genetika.

Lebih jelasnya dapat dilihat pada Gambar 2.2



Gambar 2.1. Ilustrasi Perbedaan Istilah-Istilah (Sum'ani, 2012)

Menurut Sam'ani (2012), terdapat beberapa komponen dalam Algoritma Genetika yaitu skema pengkodean, membangkitkan populasi awal dan kromosom, nilai *fitness*, seleksi, pindah silang, dan mutasi.

a. Skema Pengkodean

Teknik pengkodean adalah bagaimana mengodekan gen dari kromosom, gen merupakan bagian dari kromosom. Satu gen akan mewakili

satu variabel. Agar dapat diproses melalui algoritma genetika, maka alternatif solusi tersebut harus dikodekan terlebih dahulu kedalam bentuk kromosom. Masing-masing kromosom berisi sejumlah gen yang mengodekan informasi yang disimpan di dalam individu atau kromosom.

Gen dapat direpresentasikan dalam beberapa bentuk yaitu *bit*, bilangan *real*, *string*, daftar aturan, gabungan dari beberapa kode, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

b. Membangkitkan Populasi Awal dan Kromosom

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu atau kromosom secara acak atau melalui *procedure* tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal.

Teknik dalam pembangkitan populasi awal pada penelitian ini menggunakan metode random *search*, pencarian solusi dimulai dari suatu titik uji tertentu secara acak. Titik uji tersebut dianggap sebagai alternatif solusi yang disebut sebagai populasi.

c. Nilai *Fitness*

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran. Di dalam evolusi alam, individu yang bernilai *fitness* tinggi yang akan bertahan hidup. Sedangkan individu yang bernilai *fitness* rendah akan mati.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Nilai *fitness* digunakan untuk mengukur nilai atau derajat optimalitas suatu Kromosom. Nilai yang dihasilkan dari fungsi tersebut menandakan seberapa optimal solusi yang diperoleh. Nilai yang dihasilkan oleh *fitness* berfungsi untuk mengukur seberapa banyak jumlah pernyaratan yang dilanggar, sehingga pelanggaran pada jadwal dosen mengajar semakin kecil dan solusi yang dihasilkan semakin baik. Setiap pelanggaran yang terjadi akan diberikan nilai 1. Untuk menghindari nilai *fitness* tidak terhingga maka jumlah total semua pelanggaran akan ditambah 1. Rumus nilai *finess* seperti dibawah ini:

$$F = \frac{1}{1+(\sum BD + \sum BR + \sum WD)} \quad \text{.....2.1}$$

Keterangan :

BD = Banyaknya dosen mengajar pada waktu bersamaan

BR = Banyaknya ruangan yang digunakan pada waktu bersamaan

WD = Banyak waktu dosen yang dilanggar

d. Seleksi

Pembentukan susunan kromosom pada suatu populasi baru biasanya dilakukan secara proporsional sesuai dengan nilai *fitness*-nya. Suatu metode seleksi yang umumnya digunakan adalah *rzoulette-wheel*. Metode seleksi dengan mesin *roulette* ini merupakan metode yang paling sederhana dan sering dikenal dengan nama *stochastic sampling with replacement*.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

e. Pindah Silang (*Crossover*)

Salah satu komponen yang paling penting dalam Algoritma Genetika adalah pindah silang atau *crossover*. Sebuah kromosom yang mengarah pada solusi yang baik dapat diperoleh dari proses memindah silangkan dua buah kromosom. Pindah silang juga dapat berakibat buruk jika ukuran populasinya sangat kecil. Dalam suatu populasi yang sangat kecil, suatu kromosom dengan gen-gen yang mengarah pada solusi terbaik akan sangat cepat menyebar ke kromosom-kromosom lainnya. Untuk mengatasi masalah ini digunakan suatu aturan bahwa pindah silang hanya bisa dilakukan dengan suatu probabilitas tertentu, artinya pindah silang bisa dilakukan hanya jika suatu bilangan random yang dibangkitkan kurang dari probabilitas yang ditentukan tersebut. Pada umumnya probabilitas tersebut diset mendekati 1. Pindah silang yang paling sederhana adalah pindah silang satu titik potong (*one-point crossover*). Suatu titik potong dipilih secara acak (random), kemudian bagian pertama dari orang tua 1 digabungkan dengan bagian kedua dari orang tua pada Tabel 2.2 sebagai berikut:

Tabel 2.2 Pindah Silang Pada Algoritma Genetika (Sum'ani, 2012)

Orang tua 1	0	0	1	1	1	1	1	1	1	1	1	1
Orang tua 2	1	1	0	0	0	0	0	0	0	0	0	0
	G1			*G4*	*G5*			*G8*	*G9*			*G12*
Anak 1	0	0	0	0	0	0	0	0	0	0	0	0
Anak 2	1	1	1	1	1	1	1	1	1	1	1	1

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

f. Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Metode mutasi yang digunakan adalah mutasi dalam pengkodean nilai. Proses mutasi dalam pengkodean nilai dapat dilakukan dengan berbagai cara, salah satunya yaitu dengan memilih posisi gen bebas pada kromosom, nilai yang ada tersebut kemudian dirubah dengan suatu nilai tertentu yang diambil secara acak.

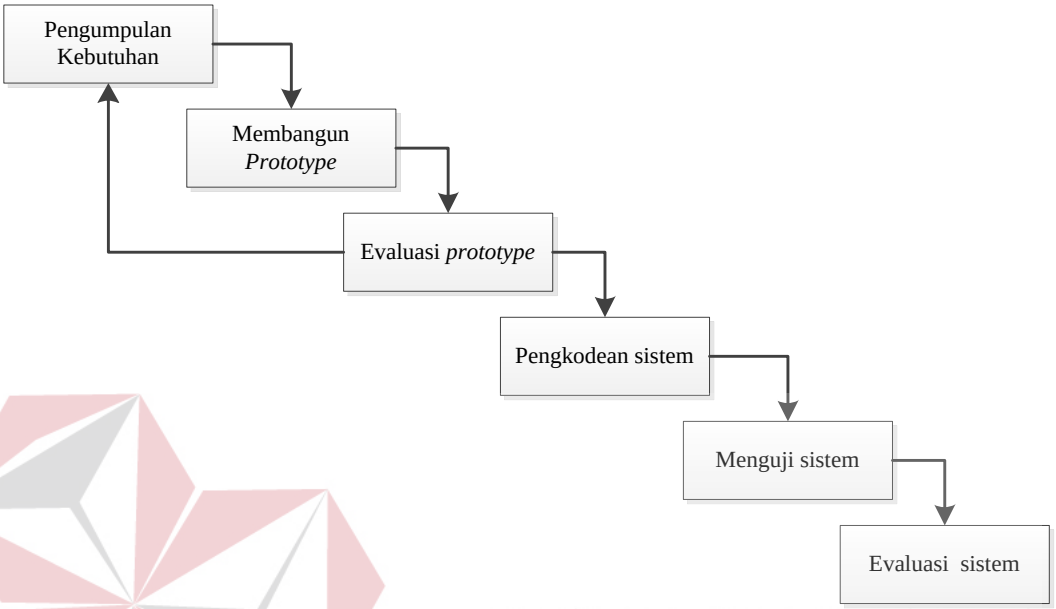
g. *Elitisme*

Karena seleksi dilakukan secara acak (*random*), maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu terpilih. Kalaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan rusak (nilai *fitness*-nya menurun) karena proses pindah silang. Untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau beberapa salinannya. Prosedur ini dikenal sebagai *Elitisme*.

2.9 Metode *System Development Life Cycle (SDLC)* Model *Prototype*

Menurut Pressman (2002), *Prototype* adalah pengembangan yang cepat dan pengujian terhadap model kerja dari aplikasi baru melalui proses interaksi dan berulang-ulang yang biasa digunakan ahli sistem informasi dan ahli bisnis. *Prototype* disebut juga desain aplikasi cepat (*rapid application design*/RAD)

karena menyederhanakan dan mempercepat desain sistem. Tahap-tahap *Prototype* menurut Pressman (2012) dapat dilihat pada Gambar 2.2.



Gambar 2.2. Tahap-tahap *Prototype* (Pressman, 2002)

Tahapan metodologi *prototype* yaitu:

a. Pengumpulan Kebutuhan

Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

b. Membangun *Prototype*

Membangun *prototype* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat format *input* dan *output*)

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

c. Evaluasi *Prototype*

Evaluasi ini dilakukan oleh pelanggan apakah *prototype* yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah 4 akan dilakukan. Jika tidak *prototype* direvisi dengan mengulangi langkah dari awal.

d. Pengkodean Sistem

Dalam tahap ini *prototype* yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

e. Menguji Sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan.

f. Evaluasi Sistem

Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika tidak langkah pengkodean sistem harus diulangi kembali

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

2.10 *Hypertext Preprocessor* (PHP)

Menurut Prasetyo (2004), PHP merupakan bahasa *scripting Server-side*, dimana pemrosesan datanya dilakukan pada sisi *Server*. Sederhananya, *Server*-lah yang akan menerjemahkan *script* program, baru kemudian hasilnya akan dikirim kepada *client* yang melakukan permintaan.

Menurut Prasetyo (2004), kelebihan PHP yaitu:

1. PHP mudah dibuat dan dijalankan, maksudnya PHP dapat berjalan dalam web *Server* dan dalam sistem operasi yang berbeda pula.
2. PHP bisa dioperasikan pada platform Linux atau Windows.
3. PHP sangat efisien, karena PHP hanya memerlukan *resource* sistem yang sangat sedikit dibanding dengan bahasa pemrograman lain.
4. Ada banyak web *Server* yang mendukung PHP, seperti Apache, PWS, IIS, dan lain-lain
5. PHP juga didukung oleh banyak *database*, seperti MySQL, postgresql, *Interbase*, SQL, dan lain-lain.
6. Bahasa pemrograman PHP menggunakan sintak yang sederhana, singkat dan mudah untuk dipahami.
7. HTML-*embedded*, artinya PHP adalah bahasa yang dapat ditulis dengan menempelkan pada sintak-sintak HTML

2.11 MySQL

Menurut Prasetyo (2004), MySQL merupakan salah satu *database Server* yang berkembang di lingkungan *open source* dan didistribusikan secara *free* (gratis) di bawah lisensi GPL. MySQL merupakan RDBMS (*Relational Database Management System*) *Server*. RDBMS adalah program yang memungkinkan pengguna *database* untuk membuat, mengelola, dan menggunakan data pada suatu *model relational*. Dengan demikian, tabel-tabel yang ada pada *database* memiliki relasi antara satu tabel dengan tabel lainnya.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Menurut Prasetyo (2004), keuntungan MySQL yaitu:

a. Cepat, handal dan mudah dalam penggunaannya

MySQL lebih cepat tiga sampai empat kali dari pada *database Server* komersial yang beredar saat ini, mudah diatur dan tidak memerlukan seseorang yang ahli untuk mengatur administrasi pemasangan MySQL.

b. Didukung oleh berbagai bahasa

Database Server MySQL dapat memberikan pesan *error* dalam berbagai bahasa seperti Belanda, Portugis, Spanyol, Inggris, Perancis, Jerman, dan Italia.

c. Mampu membuat tabel berukuran sangat besar

Ukuran maksimal dari setiap tabel yang dapat dibuat dengan MySQL adalah 4 GB sampai dengan ukuran *file* yang dapat ditangani oleh sistem operasi yang dipakai.

d. Lebih Murah

MySQL bersifat *open source* dan didistribusikan dengan gratis tanpa biaya untuk UNIX *platform*, OS/2 dan Windows *platform*.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

2.12 *Codeigniter* (CI)

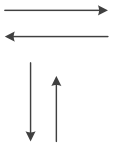
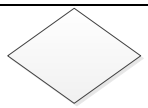
Menurut Sidik (2012), *Codeigniter* (CI) adalah *framework* pengembangan aplikasi dengan menggunakan PHP, suatu kerangka untuk bekerja atau membuat program dengan menggunakan PHP yang sistematis. Program tidak perlu membuat aplikasi dari awal karena CI menyediakan sekumpulan librari yang banyak untuk menyelesaikan pekerjaan yang umum, dengan menggunakan antarmuka dan struktur logika yang sederhana untuk mengakses librari.

Pemograman dapat memfokuskan diri pada kode yang harus dibuat untuk menyelesaikan suatu pekerjaan.

2.13 *System Flowchart*

Menurut Kristanto (2003), *System Flowchart* adalah "bagan (*chart*) yang menunjukan alir (*flow*) di dalam program atau prosedur sistem secara logika". *System Flowchart* merupakan suatu bagan yang menggambarkan arus dari data yang akan diproses dalam suatu program dari awal sampai akhir. Di dalam *system flowchart*, terdapat simbol-simbol untuk pembuatan aliran data yaitu sebagai berikut:

Tabel 2.3 Simbol *Flowchart* (Kristanto, 2003)

Simbol *Flowchart*	**Keterangan**
	Operasi secara manual
	*Input output*
	Proses
	Arus informasi
	Keputusan
	Dokumen atau laporan

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Simbol *Flowchart*	Keterangan
	Terminal
	Penyimpanan *file* secara sementara
	*Input* manual
	*Input* secara manual
	*Penghubung ke* form *berikutnya*

2.14 *Context Diagram*

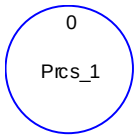
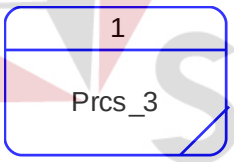
Menurut al-bahra (2006), *Context diagram* adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. *Context diagram* merupakan *level* tertinggi dari *data flow diagram* (DFD) yang menggambarkan seluruh *input* ke sistem atau *output* dari sistem. *Context diagram* akan memberi gambaran tentang keseluruhan sistem. Dalam diagram *context* hanya ada satu proses. Tidak boleh ada store dalam diagram *context*.

2.15 *Data Flow Diagram* (DFD)

Menurut Kristanto (2003), *Data Flow Diagram* (DFD) adalah suatu model data atau proses yang dibuat untuk menggambarkan aliran data dari mana asal dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut, dan interaksi antara data yang tersimpan. Dalam menggambarkan sistem perlu dilakukan pembentukan simbol,

berikut ini simbol-simbol yang digunakan dalam DFD dengan mengacu pada notasi demarco-Yourdon.

Tabel 2.4 Simbol Data *Flow* Diagram (Kristanto, 2003)

Gambar Data *Flow* Diagram	**Keterangan**
	Menunjukan entitas yang berhubungan dengan sistem yang sedang dikembangkan, dimana kesatuan luar berada diluar lingkungan sistem yang akan memeberikan *input* atau menerima *input*.
	Menunjukan arus data atau aliran data yang berupa masukan untuk sistem atau hasil dari sistem tersebut. Data *flow* juga dapat mempresentasikan data atau informasi yang tidak berkaitan dengan komputer
	Menunjukan proses kegiatan atau kerja dari fungsi transformasi komponen, dan menggambarkan bagian dari sistem mentransformasikan *input* ke *output*
	Menunjukan media penyimpanan

Menurut Kristanto (2003), di dalam DFD terdapat 3 *level*, yaitu:

1. Diagram Konteks: Diagram Konteks merupakan *level* tertinggi dari DFD, yang memperlihatkan sistem sebagai sebuah proses. Tujuannya adalah memberikan pandangan umum sistem. Diagram Konteks memperlihatkan sebuah proses yang berinteraksi dengan lingkungannya. Ada *External Entity*

yang memberikan masukan (*input*) dan ada pihak yang menerima keluaran (*output*) dari sistem.

2. Diagram Nol (diagram *level*-1): Diagram yang berada satu *level* dibawah Diagram Konteks yang menggambarkan proses-proses utama dari sistem. Hal yang digambarkan dalam diagram *Zero* adalah proses utama dari sistem serta hubungan terminator atau entitas proses, *data flow* dan *data store*.
3. Diagram Rinci: Diagram *level* n merupakan hasil dekomposisi dari *Diagram zero*, yang menjelaskan proses secara lebih terperinci. Turunan langsung dari *Diagram Zero* dinamakan Diagram *Level* 1. Dan apabila Diagram *level* 1 dapat diuraikan lagi maka akan terbentuk diagram *level* 2, dan seterusnya.

2.16 *Hierarchy Input Process Output* (HIPO)

Menurut Jogiyanto (2005), *Hierarchy Input Process Output* (HIPO) merupakan metode yang dikembangkan dan didukung oleh IBM. Tetapi saat ini HIPO banyak digunakan sebagai alat desain dan teknik dokumentasi dalam siklus pengembangan sistem atau proses-proses pada sistem".

Menurut Jogiyanto (2005), HIPO dapat digunakan sebagai alat pengembangan sistem dan teknik dokumentasi program. Penggunaan HIPO ini mempunyai sasaran utama sebagai berikut:

1. Untuk menyediakan suatu struktur guna memahami fungsi-fungsi dari program.
2. Untuk lebih menekankan fungsi-fungsi yang harus diselesaikan oleh program, bukannya menunjukan statemen-statemen program yang digunakan untuk melaksanakan fungsi tersebut.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

3. Untuk menyediakan penjelasan yang jelas dari *input* yang harus digunakan dan *output* yang harus dihasilkan oleh masing-masing fungsi pada tiap-tiap tingkatan dari diagram-diagram HIPO.
4. Untuk menyediakan *output* yang tepat dan sesuai dengan kebutuhan-kebutuhan pemakai.

2.17 *Entity Relationship Diagram* (ERD)

Menurut al-bahra (2006), ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. Jadi, jelaslah bahwa ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh sistem, sedangkan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan relationship data.

Menurut al-bahra (2006), Elemen-elemen Diagram hubungan entitas sebagai berikut:

a *Entity* (Entitas)

Pada E-R diagram digambarkan dengan bentuk persegi panjang. Entity adalah sesuatu apa saja yang ada di dalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokkan dalam empat jenis nama yaitu: orang, benda, lokasi, kejadian (terdapat unsure waktu didalamnya).

b *Relationship* (Relasi)

Pada E-R diagram digambarkan dengan sebuah bentuk belah ketupat. Relationship adalah hubungan alamiah yang terjadi antara entitas. Pada

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

umumnya penghubung (Relationship) diberi nama dengan kata kerja dasar, sehingga memudahkan untuk melakukan pembacaan relasi (bias dengan kalimat aktif atau dengan kalimat pasif).

c *Relationship Degree* (Derajat relasi)

Relationship Degree atau Derajat Relasi adalah jumlah entitas yang berpartisipasi dalam satu relationship.

d *Attribute Value*

Attribute Value atau nilai *attribute* adalah suatu occurrence tertentu dari sebuah attribute di dalam suatu entity atau relationship.

Ada dua jenis Atribut sebagai berikut:

1. Identifier (*key*) digunakan untuk menentukan suatu entity secara unik (primary key).
2. Descriptor (*nonkey attribute*) digunakan untuk menspesifikasikan karakteristik dari suatu entity yang tidak unik.

e *Cardinality* (Kardinalitas)

Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas pada entitas yang lain.

Terdapat 3 macam kardinalitas relasi yaitu:

1. *One to One*

   Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya. Berarti setiap tupel pada entitas A

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

berhubungan dengan paling banyak satu tupel pada entitas B, dan begitu pula sebaliknya.

2. *One to Many atau Many to One*

   Tingkat hubungan satu ke banyak adalah sama dengan banyak ke satu. Tergantung dari mana hubungan tersebut dilihat. Untuk satu kejadian pada entitas pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua, dan sebaliknya.

3. *Many to Many*

   Tingkat hubungan kebanyakan terjadi jika tiap kejadian pada setiap entitas akan memepunyai banyak hubungan dengan kajadian pada entitas lainnya. Baik dilihat dari sisi entitas yang pertama, maupun dilihat dari sisi yang kedua. Berarti setiap tupel pada entitas A dapat berhubungan dengan banyak tupel pada entitas B, dan demikian sebaliknya.

2.18 *Testing*

Menurut Quadri dan Farooq (2010), pengujian *software* adalah proses verifikasi dan validasi apakah sebuah aplikasi *software* atau program memenuhi persyaratan bisnis dan persyaratan teknis yang mengarahkan desain dan pengembangan dan cara kerjanya seperti yang diharapkan dan juga mengidentifikasi kesalahan yang penting yang digolongkan berdasarkan tingkat *severity* pada aplikasi yang harus diperbaiki.

Menurut Nidhra dan Dondeti (2012), pengujian software adalah teknik yang sering digunakan untuk verifikasi dan validasi kualitas suatu software.

Pengujian software adalah prosedur untuk eksekusi sebuah program atau sistem dengan tujuan untuk menemukan kesalahan.

2.19 *Black Box Testing*

Menurut Black (2009), Tester menggunakan *behavioral test* (disebut juga *Black-Box Tests*), sering digunakan untuk menemukan *bug* dalam *high level operations*, pada tingkatan fitur, profil operasional dan skenario *customer*. *Tester* dapat membuat pengujian fungsional *black box* berdasarkan pada apa yang harus sistem lakukan. *Behavioral testing* melibatkan pemahaman rinci mengenai domain aplikasi, masalah bisnis yang dipecahkan oleh sistem dan misi yang dilakukan sistem. *Behavioral test* paling baik dilakukan oleh penguji yang memahami desain sistem, setidaknya pada tingkat yang tinggi sehingga mereka dapat secara efektif Menemukan *bug* umum untuk jenis desain.

Menurut Nidhra dan Dondeti (2012), *black box testing* juga disebut *functional testing*, sebuah teknik pengujian fungsional yang merancang *Test Case* berdasarkan informasi dari spesifikasi.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA