BAB 2
TINJAUAN PUSTAKA

2.1 Profil SMA Negeri 1 Ciwidey

Profil tempat penelitian tugas akhir ini menjelaskan sejarah , logo, visi dan misi, struktur organisasi, serta deskripsi pekerjaan di sekolah menengah atas 1 Ciwidey.

2.1.1 Sejarah Instansi

SMAN 1 Ciwidey berdiri atas keinginan masyarakat Ciwidey dan sekitarnya. Harapan tersebut dapat terwujud pada tahun 1987 sebagai sekolah filial (kelas jauh) dari SMAN Soreang. Lokasi yang digunakan adalah SD Negeri Ciwidey III. Pada tahun pelajaran 1987-1988 menerima kelas 1 sebanyak 4 kelas. Kepala sekolah pada tahun tersebut adalah bapak Lili Setia BA selaku kepala SMAN 1 Soreang sekaligus sebagai pengelola SMAN 1 Ciwidey. Guru –guru di SMA Negeri 1 Ciwidey pada saat tahun tersebut 90% adalah guru dari SMAN 1 Soreang.

Tahun pelajaran 1988-1989, lokasi pindah ke SMPN 1 Ciwidey dan menerima siswa baru kelas 1 sebanyak 4 kelas.pada tahun tersebut kepala sekolah berganti oleh bapak Drs.Tadjudin Noor Rosadi , pada tahun pelajaran tersebut gutu dan staff tata usaha berjumlah 30 orang. Tahun 1989-1990,lokasi sekolah pindah ke SD Panundaan dan menerima siswa baru kelas I sebanyak 4 kelas. Pada tahun tersebut dirintis pembebasan tanah untuk lokasi SMAN Ciwidey dijalan Ciwidey-Rawabogo.

Tahun pelajaran 1990-1991 menerima siswa baru kelas I sebanyak 3 kelas. Tahun pelajaran 1991-1992 menerima siswa baru sebanyak 3 kelas, pada tahun tersebut bapak Amai Rachman menjadi kepala sekolah SMAN Soreang. Tahun pelajaran 1992-1993 menerima siswa baru kelas I sebanyak 5 kelas, pada saat itu jumlah guru dan staff tata usaha bertambah menjadi 36 orang. Tahun pelajaran 1993-1994 , tanggal 23 desember 1993 SMAN Ciwidey diresmikan sebagai SMA Negeri 1 Ciwidey, sehingga menjadi sekolah yang mandiri. Pada saat itu bapak Amai Rachman diangkat sebagai pejabat yang melaksanakan tugas (PYMT) kepala

SMA Negeri 1 Ciwidey. Pada tahun ini guru dan karyawan tata usaha bertambah menjadi 40 orang sedangkan siswa baru kelas I diterima sebanyak 5 kelas. Tahun tersebut terjadi pergantian pimpinan dari bapak Amai Rachman kepada bapak Drs.Odji S.Ar dan didampingi oleh Dra.Hj.Jajah Rumilah sebagai PLH.

Dengan perjalanan waktu dalam pembangunan SMA Negeri 1 Ciwidey tahun 2013 sampai dengan saat ini kepala sekolah dipimpin oleh bapak Taufik Rahmat M.M dengan perkembangan sekolah yang akan lebih baik.

2.1.2 Logo Instansi

Berikut adalah logo SMA Negeri 1 Ciwidey pada Gambar 2.1 :



Gambar 2.1 Logo SMA Negeri 1 Ciwidey

2.1.3 Visi dan Misi Instansi

Visi dan misi bagi sebuah instansi sekolah bisa dikatakan sebagai pedoman dan tujuan. Tanpa adanya visi dan misi sebuah instansi sekolah tidak akan bertahan dan tidak tahu apa yang harus dilakukan dilakukan oleh instansi sekolah tersebut. Berikut adalah visi dan misi SMA Negeri 1 Ciwidey :

1. Visi

   Visi dari SMA Negeri 1 Ciwidey adalah “IBADAH” yang memiliki pengertian sebagai berikut :

   I : Iman, taqwa, dan ikhlas dalam pengabdian.

   BA : Bijaksana dan professional dalam melaksanakan pembaharuan pendidikan pengajaran.

   DAH : Wadah pendidikan yang dinamis, berkualitas dalam melaksanakan program dengan berwawasan wiyatamandala.

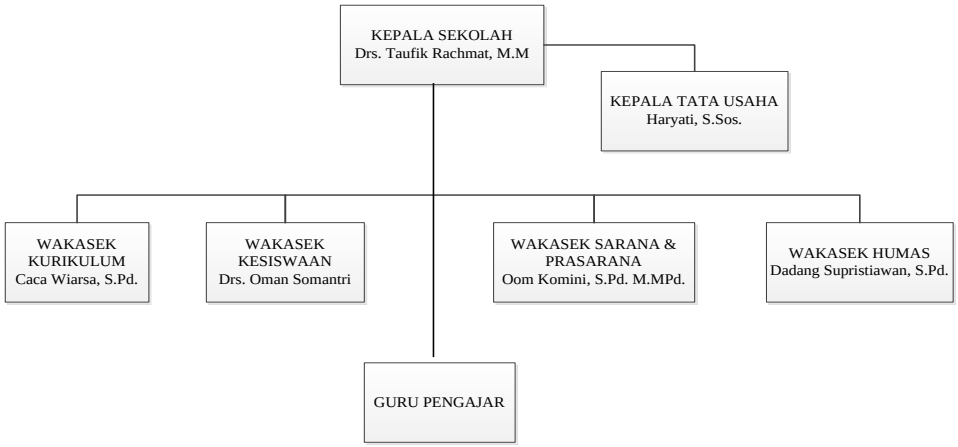
2. Misi

Misi dari SMA 1 Ciwidey adalah :

a. Membantu membentuk insan yang religius, beriman, taqwa kepada Tuhan Yang Maha Esa serta ikhlas dalam pengabdian bagi kemajuan bangsa dan negara.
b. Ikut mewujudkan tujuan pendidikan nasional yaitu dengan mencerdaskan kehidupan bangsa dalam rangka membentuk manusia Indonesia seutuhnya.
c. Mengembangkan dan meningkatkan mutu pendidikan dengan berbasis sekolah agar efektif dan efisien.
d. Membina/meningkatkan profesionalisme kinerja guru/karyawan dengan hidup di masyarakat.

2.1.4 Struktur Organisasi Instansi

Struktur organisasi SMA Negeri 1 Ciwidey dapat dilihat pada Gambar 2.2 :



Gambar 2.2 Struktur Organisasi SMA Negeri 1 Ciwidey

2.1.5 Deskripsi Pekerjaan Instansi

Berikut adalah rincian deskripsi pekerjaan dalam setiap bidang dalam sturktur organisasi di SMA Negeri 1 Ciwidey :

1. Kepala Sekolah

   Kepala sekolah merupakan pemegang wewenang utama dalam ruang lingkup sekolah. Bertugas sebagai *educator* yang melaksanakan bimbingan kepada guru dalam menyusun dan melaksanakan program pengajaran. Sebagai pemegang wewenang utama, kepala sekolah memiliki integritas serta bertanggung jawab atas lingkungan sekolah.

2. Kepala Tata Usaha

   Kepala tata usaha merupakan bagian yang melakukan pengolahan seluruh data administrasi sekolah yang bekerja sama dengan kepala sekolah dalam membantu untuk mengurusi dan mengawasi seluruh administrasi. Pekerjain lain yang dilakukan kepala tata usaha antara lain :

   a. Membantu proses belajar mengajar
   b. Membantu urusan kesiswaan, kepegawaian, sarana dan prasana, pembuatan surat-surat, arsip sekolah, keuangan, serta infrastruktur sekolah
   c. Melakukan komunikasi baik antar sekolah maupun departemen pendidikan terkait.

3. Wakasek Kurikulum

   Wakil kepala sekolah bidang kurikulum bertanggung jawab terhadap kelancaran kegiatan belajar mengajar disekolah. Tugas wakasek kurikulum, diantaranya :

   a. Menyusun program pengajaran.
   b. Menyusun pembagian tugas guru
   c. Menyusun kalender pendidikan dan time schedule kegiatan.
   d. Menyusul jadwal pelajaran.
   e. Menyusun jadwal evaluasi belajar.
   f. Menyusun pelaksanaan ujian akhir.
   g. Menetapkan kriteria penyataan naik kelas/tidak naik kelas.
   h. Menerapkan jadwal penerimaan buku laporan pendidikan (raport) dan penerimaan STTTB/Ijazah.
   i. Mengkoordinasikan dan mengarahkan dalam penyusunan materi pembelajaran.

j. Menyediakan buku kenaikan kelas.
k. Memonitor jalannya KBM.
l. Mendorong prestasi akademik siswa.
m. Mengadakan supervisi kelas.
n. Menyusun laporan pelaksanaan pelajaran/target kurikulum.

4. Wakasek Kesiswaan

Wakil kepala sekolah bidang kesiswaan bertanggung jawab terhadap kegiatan yang berkaitan dengan kesiswaan. Tugas wakasek kesiswaan, diantaranya :

a. Menyusun program pembinaan kesiswaan/OSIS
b. Melaksanakan bimbingan pengarahan dan pengendalian kegiatan siswa/OSIS dalam rangka menegakkan disiplin dan tata tertib sekolah.
c. Membina dan melaksanakan koordinasi keamanan, kebersihan, ketertiban, keindahan, kerindangan, dan kekeluargaan.
d. Memberikan pengarahan dalam pemilihan pengurus OSIS.
e. Melakukan pembinaan pengurus OSIS dan berorganisasi.
f. Menyusun program dalam pembinaan siswa secara berkala dan insidential.
g. Mengkoordinasikan pemilihan calon siswa teladan dan calon siswa penerima beasiswa.
h. Mengadakan pemilihan siswa untuk mewakili sekolah dalam kegiatan di luar sekolah.
i. Mengatur mutasi siswa.
j. Menyusun laporan pelaksanaan kegiatan kesiswaan secara berkala.

5. Wakasek Sarana dan Prasana

Wakil kepala sekolah bidang sarana dan prasarana bertanggung jawab terhadap ketersedian sarana prasarana dan bahan ajar yang mendukung keterlaksanaanya kegiatan belajar mengajar. Tugas wakasek sarana dan prasarana, diantaranya :

a. Menyusun rencana kerja dan jadwal kerja.
b. Mengkoordinasikan pendayagunaan sarana dan prasarana sekolah.

c. Mengkoordinasikan kegiatan perencanaan dan pengadaan bahan-bahan pengajaran pengajaran.
d. Mengkoordinasikan kegiatan program perbaikan dan pemeliharaan sarana/prasarana sekolah.
e. Mengkoordinasikan kegiatan pembangunan / penambahan sarana dan prasarana baru di lingkungan sekolah

6. Wakasek Humas

Wakil kepala sekolah bidang humas bertanggung jawab terhadap kegiatan yang terkait dengan kerjasama dunia usaha/dunia industry, masyarakat, dan penelusuran tamatan. Tugas wakasek humas, diantaranya :

a. Menyusun Program yang terkait dengan kegiatan kehumasan
b. Mengadakan MOU dengan dunia usaha / dunia industry.
c. Mengkoordinir pelaksanaan praktek industri.
d. Mengkoordinir pelaksanaan penelusuran tamatan.
e. Mengkoordinir pelaksanaan kegaiatan unit produksi.
f. Mengkoordinir pelaksanaan magang di indsutri bagi guru.
g. Melakukan koordinasi dengan komite sekolah.
h. Mengkoordinir pelaksanaan kegiatan pemasaran tamatan.
i. Mengkoordinir pelaksanaan kegiatan promosi sekolah.

7. Guru Pengajar

Guru pengajar bertugas sebagai pelaku atau subjek dalam proses kegiatan belajar mengajar yang memberikan pendidikan serta pengajaran kepada siswa dalam rangka melaksanakan kegiatan belajar mengajar disekolah.

2.2 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dari teori-teori yang berkaitan dengan implementasi algoritma genetika pada penjadwalan. Beberapa teori yang berkaitan dengan implementasi algoritma genetika pada penyusunan penjadwalan ini adalah teori algoritma genetika, penjadwalan , serta membahas tentang basis data ,UML, c# ,visual studio ,MySql , serta xampp.

2.2.1 Algoritma Genetika

Algoritma genetika merupakan suatu metode heuristic yang dikembangkan berdasarkan prinsip genetika dan proses seleksi alamiah Teori Evolusi Darwin. Metode optimasi dikembangkan oleh John Holland sekitar tahun 1960-an dan dipopulerkan oleh salah seorang mahasiswanya, David Goldreg pada tahun 1980-an. Proses pencarian penyelesaian atau proses terpilihnya suatu individu untuk bertahan hidup dalam proses evolusi.

Dalam teori evolusi Darwin, suatu individu tercipta secara acak kemudian berkembang biak melalui proses reproduksi sehingga terbentuk sekumpulan individu sebagai populasi. Setiap individu dalam populasi mempunyai tingkat kebugaran yang berbeda-beda. Tingkat kebugaran ini menentukan seberapa kuat untuk tetap betahan hidup dalam populasinya. Sebagian individu tetap bertahan hidup dan sebagian lainnya mati.

Pada dasarnya, algoritma genetika merupakan metode pencarian yang didasarkan pada proses evolusi alamiah, yaitu terbentuknya populasi awal secara acak yang terdiri dari individu-individu dengan sifat yang bergantung pada gen-gen dalam kromosomnya. Individu-individu melakukan proses reproduksi untuk melahirkan keturunan. Sifat keturunan dibentuk dari kombinasi sifat kedua induknya atau mewarisi sifat-sifat induknya [1].

2.2.1.1 Definisi Penting Dalam Algoritma Genetika

Dalam algoritma genetika terdapat elemen-elemen yang digunakan sebagai representas sebuah variable, elemen tersebut saling berkaitan serta mempunyai peranan masing-masing , berikut adalah penjelesan elemen-elemen tersebut [4]:

1. *Allele*

   Allele merupakan suatu nilai yang terdapat pada suatu gen. Satu gen mempunyai satu nilai yang disebut *allele*. *Allele* dapat direpresentasikan dalam notasi yang telah ditentukan (biner , float, integer, maupun karakter).

2. Gen

   Gen merupakan nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Gen mewakili

satu atau lebih variabel sebagai solusi yang ingin dicapai. Gen dapat di representasikan dalam bentuk berikut :

String bit : 10011, 01101, 11101, dst.

Bilangan real : 65.65, -67.98, 592.88, dst.

Elemen permutasi : E2, E10, E6, dst.

Daftar aturan : R1, R2, R3, dst.

Elemen program : pemograman genetika.

3. Kromosom atau Individu

   Kromosom atau individu merupakan gabungan dari kumpulan gen-gen yang membentuk nilai tertentu dan menyatakan solusi yang mungkin dari suatu permasalahan.

4. Populasi

   Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu satuan siklus evolusi.

5. *Fitness* (Nilai *fitness*)

   Fitness merupakan suatu nilai yang menyatakan seberapa baik suatu individu yang didapatkan. Semakin besar nilai *fitness* semakin baik suatu individu untuk bertahan hidup.

6. Seleksi

   Seleksi merupakan proses pemilihan dua individu terbaik/ memiliki nilai *fitness* yang tinggi yang akan dijadikan sebagai induk (*parent*) untuk digunakan dalam proses regenerasi. Seleksi juga berfungsi memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit.

7. *Crossover*

   Crossover merupakan proses pertukaran atau kawin silang gen-gen dari dua induk tertentu.

8. Mutasi

   Mutasi merupakan proses pergantian salah satu gen yang terpilih dengan nilai tertentu.

9. Generasi

   Generasi merupakan urutan iterasi dalam satu kesatuan proses genetika, dimana beberapa kromososm terbaik dari generasi sebelumnya bergabung dan membuat kromosom baru dengan solusi yang lebih baik.

10. *Offspring*

    Offspring merupakan kromosom baru yang dihasilkan setelah melewati suatu generasi. Dalam hal ini *offspring* tersebut merupakan hasil kromosom yang dibangkitkan setelah melakukan proses seleksi.

2.2.1.2 Aplikasi Algoritma Genetika

Sejak pertama kali dirintis oleh John Holland pada tahun 1960-an, algoritma genetika telah dipelajari, diteliti dan diaplikasikan secara luas pada berbagai bidang. Algoritma banyak digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter optimal [5].

Keuntungan penggunaan algoritma genetika sangat jelas terlihat dari kemudahan implementasi dan kemampuannya untuk menentukan solusi seperti:

1. Ruang masalah sangat besar, kompleks, dan sulit dipahami.
2. Kurang atau bahkan tidak ada pengetahuan yang memandai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit.
3. Tidak tersedianya analisis matematika yang memadai.
4. Ketika metode-metode konvensional sudah tidak mampu menyelesaikan masalah yang dihadapi.
5. Solusi yang diharapkan tidak harus paling optimal, tetapi cukup bisa diterima.
6. Terdapat batasan waktu, misalnya dalam real time systems atau system waktu nyata.

Algoritma genetika telah banyak diaplikasikan untuk penyelesaian masalah dan pemodelan dalam bidang teknologi, bisnis dan entertainment seperti:

1. Optimasi

   Algoritma Genetika untuk optimasi *numeric* dan optimasi kombinatorial seperti *Traveling Salesman Problem* (TSP), perancangan *Intergrated Circuit* atau IC, *job shop scheduling*, optimasi video, dan suara.

2. Pemograman otomatis

   Algoritma genetika telah digunakan untuk melakukan proses evolusi terhadap program komputer untuk merancang struktur komputasional, seperti *cellular automatic* dan *sorting networks*.

3. *Machine learning*

   Algoritma genetika telah berhasil diaplikasikan untuk memprediksi struktur protein. Algoritma genetika juga berhasil diaplikasikan dalam perancangan neural networks (jaringan syaraf tiruan) untuk melakukan proses evolusi terhadap aturan-aturan pada *learning classifier systems* atau *symbolic prosuction systems*. Algoritma genetika juga digunakan untuk mengontrol robot.

4. Model Ekonomi

   Algoritma genetika telah digunakan untuk memodelkan proses-proses inovasi dan pembangunan bidding strategies.

5. Model Sistem Imunisasi

   Algoritma genetika telah berhasil digunakan untuk memodelkan berbagai aspek pada sistem imunisasi alamiah, termasuk somatic mulation selama kehidupan individu dan menentukan keluarga dengan gen ganda (multi - gen families) sepanjang waktu evolusi.

6. Model Ekologi

   Algoritma genetika telah berhasil digunakan untuk memodelkan fenomena ekologis seperti *host-parasite co-evolutions*, simbiosis dan aliran sumber daya dalam ekologi.

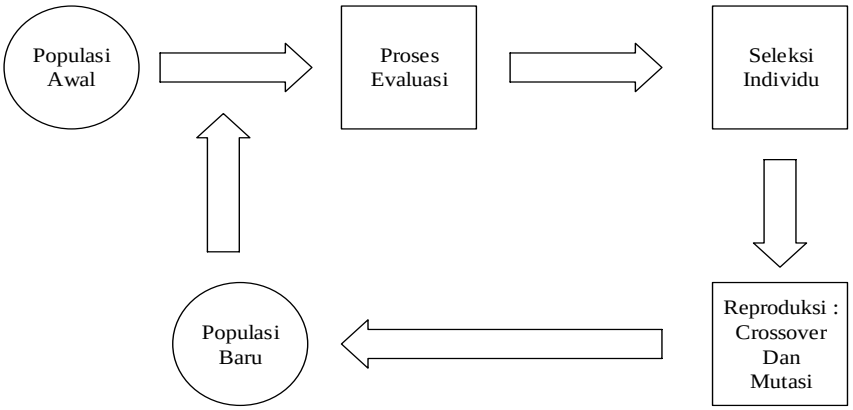
7. Interaksi antara Evolusi dan Belajar

   Algoritma genetika telah digunakan untuk mempelajari bagaimana proses belajar suatu individu bisa mempengaruhi proses evolusi suatu species dan sebaliknya.

2.2.1.3 Siklus Algoritma Genetika

Siklus algoritma genetika dibutuhkan pada setiap kasus, siklus petama kali memperkenalkan siklus adalah oleh David Goldberg. Siklus dimulai dari membuat

populasi awal secara acak, kemudian setiap individu dihitung nilai *fitness*-nya. Proses berikut adalah menyeleksi individu terbaik, kemudian dilakukan *crossover* dan dilanjutkan oleh proses mutasi sehingga terbentuk populasi baru. Selanjutnya populasi baru ini mengalami siklus yang sama dengan populasi sebelumnya. Proses tersebut berlangsung hingga generasi ke –n. Berikut siklus algoritma genetika menurut David Goldberg pada Gambar 2.3 [6] :



Gambar 2.3 Siklus Algoritma Genetika David Goldberg

Siklus algoritma genetika pada Gambar 2.3 akan menjadi acuan untuk siklus implementasi algoritma genetika pada penjadwalan mata pelajaran.

2.2.1.4 Komponen Utama Algoritma Genetika

Komponen-komponen utama algoritma genetika saling berketergantungan dan berkaitan satu sama lain, berikut penjelasan dari komponen tersebut [3]:

2.2.1.4.1 Teknik Pengkodean

Teknik pengkodean merupakan bagian penting dalam algortima genetika. Proses ini diperlukan dalam kaitan dengan peranan kromosom sebagai representasi penyelesaian masalah. Kromosom gabungan dari gen-gen yang membentuk nilai tertentu. Satu gen akan mewakili satu *variable*, agar dapat diproses melalui algoritma genetika. Gen dapat direpresentasikan dalam bentuk : string bit, bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

2.2.1.4.2 Membangkitkan Populasi Awal

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

Teknik dalam pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah sebagai berikut :

1. Random search

   Pencarian solusi dimulai dari suatu titik uji tertentu. Titik uji tersebut dianggap sebagai alternatif solusi yang disebut sebagai populasi.
2. Random Generator

   Random generator melibatkan pembangkitan bilangan random untuk nilai setiap gen sesuai dengan representasi kromosom yang akan digunakan.
3. Pendekatan tertentu

   Cara ini dengan memasukkan nilai tertentu kedalam gen dari populasi awal yang dibentuk.
4. Permutasi Gen

   Cara ini dengan penggunaan permutasi josephus dalam kombinatorial.

Pembentukan populasi awal yang digunakan adalah *random search*, dengan pencarian solusi berdasarkan satu titik uji yaitu dari data tugas mengajar.

2.2.1.4.3 Nilai Fitness

Suatu individu atau kromosom dievaluasi berdasarkan suatu fungsi tertentu sebagai ukutan performainya. Didalam evolusi alam, individu yang bernilai *fitness* tinggi yang akan bertahan hidup. Sedangkan bernilai *fitness* rendah akan mati.

Pada masalah optimasi, jika solusi yang dicari adalah memaksimalkan sebuah fungsi h (dikenal sebagai masalah maksimasi), maka nilai *fitness* yang digunakan adalah nilai dari fungsi h tersebut, yakni f = h (dimana f adalah nilai *fitness*). Tetapi jika masalahnya adalah meminimalkan fungsi h, maka fungsi h tidak bisa digunakan secara langsung. Hal ini disebabkan adanya aturan bahwa individu

yang memiliki nilai *fitness* tinggi lebih mampu bertahan hidup pada generasi berikutnya. Oleh karena itu nilai *fitness* yang digunakan adalah *f* = 1/h , yang artinya semakin kecil nilai h, semakin besar nilai f . Tetapi hal ini akan menjadi masalah jika h bisa bernilai 0, yang mengakibatkan f bernilai tak terhingga. Untuk mengatasinya, h perlu ditambah sebuah bilangan yang dianggap sangat kecil sehingga nilai *fitness* dapat dilihat pada persamaan 1 :

$$f = \frac{1}{(h + a)} \tag{2.1}$$

dimana a adalah bilangan yang dianggap sangat kecil dan bervariasi sesuai dengan masalah yang akan diselesaikan. Penentuan nilai *fitness* sangat berpengaruh pada performasi algoritma genetika secara keseluruhan.

Nilai *fitness* yang digunakan untuk mencapai nilai optimal dalam penelitian ini adalah 1. Hasil nilai fitness tersebut adalah dari aturan yang telah ditetapkan dengan memberikan nilai 0 untuk aturan yang tidak dilanggar dan 1 untuk aturan yang dilanggar. Maka bila dimasukan pada rumus persamaan 1 akan menghasilkan nilai *fitness* 1, dengan h adalah mengenai aturan-aturan yang telah ditetapkan dan a adalah nilai terkecil yang digunakan adalah 1.

2.2.1.4.4 Seleksi

Setiap kromosom yang terdapat dalam populasi akan melalui proses seleksi untuk dipilih menjadi orang tua. Ada beberapa metode seleksi , diantaranya :

1. Seleksi Roulette Wheel

   Metode seleksi yang umum digunakan adalah seleksi *roulette wheel* , sesuai dengan namanya, metode ini menirukan permainan *roulette wheel*. Dimana masing-masing kromosom menempati potongan lingkaran pada roda *roulette wheel* secara proporsional sesuai dengan nilai *fitness*nya. Kromosom yang memiliki nilai *fitness* lebih besar menempati potongan lingkaran yang lebih besar dibandingkan kromosom bernilai *fitness* rendah.

2. Seleksi Ranking

   Seleksi ini memperbaiki proses seleksi yang sebelumnya yaitu *roulette wheel* karena pada seleksi tersebut kemungkinan selain satu kromosom

mempunyai nilai fitness yang mendominasi hingga 90%. Nilai fitness yang lain akan mempunyai kemungkinan yang sangat kecil untuk terpilih. Seleksi rangking dipakai untuk mengatasi masalah dari kekurangan seleksi *roulette wheel*. Pertama-tama, diurutkan seluruh kromosom berdasarkan bagus-tidaknya solusi berdasarkan nilai *fitness*-nya. Setelah diurutkan, kromosom terburuk diberi nilai fitness baru sebesar 1, kromosom kedua terburuk diberi nilai fitness baru sebesar 2, dan seterusnya. Kromosom terbaik diberi nilai fitness baru sebesar n dimana n adalah banyak kromosom dalam suatu populasi.

3. Seleksi Steady State

   Metode ini tidak banyak digunakan dalam proses seleksi karena dilakukan dengan mempertahankan individu yang terbaik. Pada setiap generasi, akan dipilih beberapa kromosom-kromosom yang memiliki nilai *fitness* terburuk akan digantikan dengan *offspring* yang baru. Sehingga pada generasi selanjutnya akan terdapat beberapa populasi yang dipertahankan.

4. Seleksi Turnamen

   Merupakan metode seleksi lainnya yang didasari fenomena alamiah seperti turnamen antar individu dalam populasi. Dilakukan dengan memilih secara acak beberapa kromosom dari populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk.

5. Truncation Random

   Metode ini lebih mudah diterapkan jika dibandingkan dengan metode *roulette wheel*, pemilihan kromosom dilakukan secara acak tetapi tidak semua kromosom mendapatkan kesempatan tersebut, hanya kromosom terbaik saja yang berpeluang.

Seleksi yang digunakan pada penelitian ini adalah seleksi rank. Seleksi rank adalah perbaikan dari seleksi *roulette wheel* yang sering digunakan dalam penelitian mengenai penjadwalan. Selain itu seleksi *rank* dipilih berdasarkan saran dari beberapa penelitian mengenai percobaan menggunakan metode seleksi yang lain.

2.2.1.4.5 Crossover (Pindah Silang)

Sebuah kromosom yang mengarah pada solusi yang bagus dapat diperoleh dari proses memindah-silangkan dua buah kromosom. Pindah silang dapat berakibat buruk jika ukuran populasinya sangat kecil. Dalam suatu populasi yang sangat kecil, suatu kromosom dengan gen-gen yang mengarah ke solusi akan sangat cepat menyebar ke kromosom-kromosom lainnya. Untuk mengatasi masalah tersebut digunakan sebuah aturan bahwa pindah silang hanya dapat dilakukan dengan sebuah probabilitas tertentu $p_{c.}$ Pindah silang dapat dilakukan hanya jika suatu bilangan random [0,1] yang dibangkitkan kurang dari p_c yang ditentukan. Pada umumnya p_c diset mendekati 1 , misalnya 0,8.

Ada beberapa cara yang dapat digunakan untuk melakukan pindah silang sesuai dengan teknik pengkodean (*encoding)* yaitu sebagi berikut [1] :

1. *Binary encoding*

a. Crossover satu titik

   Memilih satu titik tertentu, selanjutnya nilai biner sampai titik crossovernya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai biner dari induk kedua.

b. *Crossover* N-titik

   Pada metode ini setiap kromosom induk dipotong menjadi N+1 bagian. Kromosom anak yang pertama dihasilkan dengan mewariskan bagian potongan urutan ganjil dari induk yang pertama dan mewariskan bagian potongan dalam urutan genap dari induk kedua. Demikian juga dengan kromosom anak yang kedua dihasilkan dengan cara yang sama dengan menggabungkan bagian yang tersisa dari kedua kromosom induknya.

c. *Crossover uniform*

   Pada metode penyilangan ini, perlu dibangkitkan terlebih dahulu topeng penyilangan (*crossover mask*) yang terdiri atas kode biner sebanyak gen dalam kromosom. Gen-gen dalam kromosom anak ditentukan berdasarkan kode biner topeng penyilang. Untuk kromosom anak yang pertama, jika kode pada kedudukan topeng penyilangan bernilai satu, maka gen diwariskan dari induk

pertama, sedangkan jika induk kode pada topeng penyilangan bernilai nol, maka gen diwariskan dari induk kedua.

2. *Permutation encoding*

   Memilih satu titik tertentu, nilai permutation sampai titik crossover. Pada induk pertama digunakan lalu sisanya dilakukan scan terlebih dahulu, jika nilai permutasi pada induk kedua belum ada pada *offspring* nilai tersebut ditambahkan.

3. *Value encoding*

   Metode *crossover* pada pada pengkodean biner dapat digunakan.

4. *Tree encoding*

   Memilih satu titik tertentu dari tiap induk, dan menggunakan tree dibawah titik pada induk pertama dan tree dibawah induk kedua.

2.2.1.4.6 Mutasi

Mutasi merupakan operator dalam algoritma genetika yang bertujuan untuk mengubah gen-gen tertentu dari sebuah kromosom. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Beberapa cara operasi mutasi yang diterapkan dalam algoritma genetika, antara lain :

a. Mutasi dalam pengkodean integer

   Mutasi pada pengkodean integer harus dipastikan dapat menjaga konsistensi urutan nilai kromosomnya. Mutasi untuk jenis pengkodean ini dilakukan dengan cara memilih dua posisi pada kromosom, kemudian nilainya saling dipertukarkan.

b. Mutasi dalam pengkodean biner

   Mutasi pada pengkoden biner merupakan operasi yang sangat sederhana. Proses yang dilakukan yaitu dengan mengubah nilai gen pada posisi tertentu. Jika bilangan random yang dibangkitkan kurang dari propabilitas mutasi p_{mut} yang ditentukan maka ubah gen tersebut gen tersebut menjadi nilai kebalikkannya (pada pengkodean biner, 0 diubah 1 , dan 1 diubah 0). p_{mut} diset sebagai $1/n$, dimana n adalah jumlah gen dalam kromosom.

c. Mutasi dalam pengkodean permutasi

Proses mutasi yang dilakukan pengkodean biner dengan mengubah langsung bit berbeda dengan yang dilakukan pada pengkodean permutasi. Pengkodean permutasi karena yaitu metode berbasis posisi, mutasi berbasis urutan dan mutasi campur aduk. Salah satu cara yang dapat dilakukan metode berbasis posisi dilakukan dengan cara memilih posisi sebuah gen secara acak, kemudian gen tersebut akan dipindahkan pada posisi acak lainnya.

2.2.1.4.7 Syarat Berhenti

Proses optimasi yang dilakukan oleh algoritma genetika akan berhenti setelah suatu syarat berhenti terpenuhi. Beberapa syarat berhenti yang biasa digunakan adalah batas nilai fungsi *fitness*, batas waktu komputasi, banyak generasi dan terjadinya konvergensi. Syarat berhenti yang sering digunakan adalah banyak generasi, tetapi tidak menutup kemungkinan untuk dipilih kombinasi beberapa syarat berhenti [4].

2.2.1.4.8 Penentuan Parameter Algoritma

Parameter algoritma merupakan salah satu bagian penting dalam penerapan algoritma gentika yang tidak mudah untuk ditentukan secara pasti. Tidak ada aturan yang pasti untuk menentukan parameter algoritma, baik probabilitas *crossover* (Pc), probabilitas mutasi (Pm), maupun ukuran populasi. Hal ini tidak terlepas dari prinsip algoritma genetika yang mengandalkan bilangan acak hampir dalam setiap langkahnya, mulai dari pembentukan populasi awal, proses penyilangan atau proses mutasi. Bilangan acak yang berbeda pasti akan menyebabkan hasil yang berbeda pula [3]. Ada beberapa rekomendasi yang dapat digunakan , antara lain :

a. Untuk permasalahan yang memiliki kawasan solusi cukup besar , De Jong merekomdasikan untuk nilai parameter kontrol :

(Popsize ; Pc ; Pm) = (50 ; 0,6; 0,001)

b. Bila rata-rata *fitness* setiap generasi digunakan sebagai indikator, maka Grefensttete merekomendasikan :

(Popsize ; Pc ; Pm) = (30 ; 0,95; 0,01)

c. Bila *fitness* dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah :

(Popsize ; Pc ; Pm) = (80 ; 0,45; 0,01)

d. Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

Pada penelitian ini rekomendasi yang digunakan adalah rekomendasi c , dipilih rekomendasi c dikarenakan untuk mencapai tujuan yang diinginkan dengan *fitness* = 1 ,harus dipantau setiap generasi. Maka dipilih rekomendasi c dengan Popsize (ukuran populasi) 80, Pc (probabilitas *crossover*) 0,45. Untuk ukuran populasi tidak digunakan dikarenakan telah menggunakan banyak pengecekan serta ingin mencapai tujuan dengan *fitness* = 1 .

2.2.2 Penjadwalan

Penjadwalan adalah proses pengurutan pembuatan produk secara menyeluruh pada beberapa mesin. Penjadwalan dapat pula dipandang sebagai proses pengalokasian sumber daya pada jangka waktu tertentu untuk melakukan sekumpulan tugas [7].

Definisi penjadwalan ini mengandung dua arti yang berbeda, yaitu :

1. Penjadwalan merupakan fungsi pengambilan keputusan, yakni untuk menentukan jadwal (nilai praktis).
2. Penjadwalan merupakan suatu teori, yakni sekumpulan prinsip-prinsip dasar, model-model, teknik-teknik, dan kesimpulan logis yang memberikan pengertian dalam fungsi penjadwalan.

Persoalan penjadwalan timbul apabila beberapa pekerjaan (*job*) akan dikerjakan bersamaan, sedangkan sumber daya seperti mesin atau peralatan yang dimiliki jumlahnya terbatas. Untuk mencapai hasil yang optimal dengan keterbatasan sumber daya yang dimiliki, maka diperlukan adanya penjadwalan sumber-sumber tersebut secara efisien.

2.2.3 Basis Data

Perancangan basis data merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Penerapan basis data dalam sistem informasi disebut dengan sistem basis data.

Basis data terdiri dari dua kata yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang / berkumpul, sedangkan data adalah respresentasi faktu dunia nyata yang mewakili suatu objek seperti manusia, pegawai, siswa, pembeli, pelanggan, barang, hewan, dan sebagainya yang diwujudkan dalam bentuk angka, simbol, huruf, teks, gambar, bunyi atau kombinasi sebagai satu kesatuan istrilah, Basis Data (Database) dapat didefinisikan dalam sejumlah sudut pandang seperti :

1. Himpunan kelompok data (Arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi)
3. Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Jika ditarik kesimpulan dari definisi diatas, maka basis data adalah kumpulan data yang saling berhubungan dan memudahkan dalam pengambilan data kembali. Sedangkan sistem basis data adalah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi dan tuga khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses tertentu. Proses yang memungkinkan memasukkan dan mengambil data dari media penyimpanan data dengan menggunakan perangkat lunak yang disebut dengan sistem manajemen basis data (DBMS).

2.2.4 Entity Realitionship Diagram (ERD)

Entity Realitionship Diagram (ERD) merupakan suatu diagram yang digunakan untuk menggambarkan hubungan antara entitas yang terdapat dalam

sistem. ERD adalah sebuah diagram yang menggambarkn model relasi antar rancangan data tersimpan. Model relasi ini diperlukan untuk menggambarkan struktur dari data relasi antar data. Adapun elemen – elemen ERD yaitu sebagai berikut [8] :

1. Entitas

   Entitas adalah segala sesuatu yang dapat digambarkan oleh data. Entitas juga dapat diartikan sebagai individu yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Ada dua macam entitas yaitu entitas kuat dan entitas lemah. Entitas kuat merupakan entitas yang tidak memiliki ketergantungan dengan entitas lainnya, sedangkan entitas lemah merupakan entitas yang kemunculannya tergantungan pada keberadaan entitas lain dalam suatu relasi.

2. Atribut

   Antribut merupakan pendeskripisan karakteristik dari entitas. Nilai atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu entity atau relationship. Jenis-jenis dari atribut diantaranya :

   a. *Key*
   b. *Atribut Simple*
   c. *Atribut Multivalue*
   d. *Atribut Composite*
   e. *Atribut Derivatif*

3. Relasi

   Relasi menunjukan hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Terdapat empat jenis kardinalitas relasi yang terdapat pada ERD :

   a. Relasi banyak ke satu (N-1)
   b. Relasi banyak ke banyak (N-N)
   c. Relasi satu ke banyak (1-N)
   d. Relasi satu ke satu (1-1)

2.2.5 Pemograman Berorientasi Objek (PBO)

Pemograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Menggunakan pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpuln objek yang berkorespondensi dengan objek dunia nyata.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman, dan pengujian perangkat lunak. Berikut adaah beberapa konsep dasar yang harus dipahami pemograman berorientasi objek [12] :

1. Kelas (*Class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*Method)*

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.

4. Atribut (*Attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek

dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapulasi (*Encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.

7. Pewarisan (*Inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

   Antarmuka atau *interface* sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi antar objek

    Komunikasi antar-objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*Polymorphism*)

    Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

Package adalah sebuah kontainer atau kemasan yang daoat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.2.5.1 Unified Modeling Language (UML)

UML singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak. Berikut diagram-diagram tersebut [9]:

1. Use Case Diagram

Use case diagram merupakan pemodelan untuk tingkah laku (*behavior)* pada sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case* diagram digunakan untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case* diagram yaitu aktor dan *use case*.

a. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
b. *Use Case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem.

2. Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak. Setiap *use case* yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*. *Activity diagram* juga banyak digunakan untuk mendefinisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem.
b. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan tampilan antarmuka.

c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu.

Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari class. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [12].

4. Class Diagram

Class diagram menunjukkan interaksi dan relasi antar class yang ada di dalam sistem. Sebuah class memiliki nama class, attribut, dan methods :

a. Nama class merupakan nama dari sebuah class yang digunakan dalam sebuah sistem.
b. Atribut merupakan variabel-variabel yang dimiliki oleh suatu class.
c. *Methods* merupakan fungsionalitas atau pekerjaan yang dilakukan oleh suatu class.

Atribut dan methods dapat memiliki salah satu sifat berikut :

a. *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
c. *Public,* dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar class dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam class diagram :

a. Asosiasi

   Asosiasi merupakan hubungan antar *class* yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.

b. Agregasi

   Agregasi merupakan relasi yang membuat class yang saling berelasi terikat satu sama lain namun tidak terlalu berkegantungan.

c. Composition

   Composition merupakan relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.

d. Dependency

   Dependency merupakan hubungan antar class dimana class yang memiliki relasi dependency menggunakan class lain sebagai atribut pada method.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau *programmer* dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut [12] :

a. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan pada *user*.

c. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

d. Kelas yang diambil dari pendefinisian data (model)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut *multivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada didalam perancangan kelas.

2.2.6 C#

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan.

2.3 Perangkat Lunak Pendukung

Perangkat lunak pendukung merupakan perangkat lunak yang digunakan dalam mendukung pembangunan perangkat lunak. Berikut penjelasan dari perangkat lunak pendukung :

2.3.1 Visual Studio 2010

Visual Studio 2010 merupakan alat untuk membangun sistem yang dibangun oleh Microsoft dengan berbasis .NET Framework. Visual studio 2010 dapat digunakan sebagai alat pengembangan perangkat lunak dan sistem berbasis desktop, mobile, dan web.

2.3.2 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. Tidak seperti PHP atau Apache yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu MySQL AB. MySQL AB

memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius [10].

2.3.3 Xampp

XAMPP merupakan singakatan dari X (program yang dapat dijalankan diberbagai sistem operasi), Apache, MySQL, PHP, Perl. XAMPP merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket. Terdapat Apache (web server), MySQL (database), PHP (server side scripting), Perl, FTP Server, phpMyAdmin dan berbagai pustaka bantu lainnya. Dengan menginstall XAMPP tidak perlu melakukan instalasi dan konfigurasi web server Apache, PHP dan MySQL secara manual. XAMPP akan menginstalasi dan mengkofigurasikannya secara otomatis.

2.4 Pengujian Aplikasi

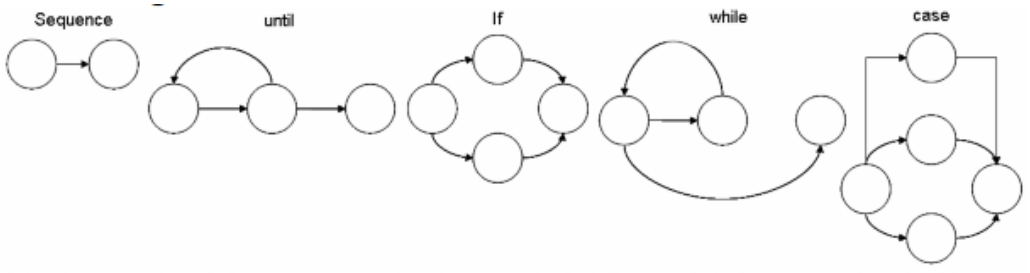
Pengujian bertujuan untuk mencari kesalahan. Pengujian yang baik adalah pengujian yang memiliki kemungkinan besar dalam menentukan kesalahan [11]. Berikut adalah penjelasan mengenai pengujian :

2.4.1 White Box Testing (Pengujian Kotak Putih)

Pengujian *white box* didasarkan pada pemeriksaan yang teliti terhadap detail prosedural. Jalur logis diseluruh perangkat lunak dan kolaborasi antar-komponen diuji dengan menguji serangkaian kondisi dan atau *loop* spesifik. Pengujian *white box* akan membawa pada kebeneran program 100 persen, yang perlu dilakukan adalah mendefinisikan semua jalur logis, mengembangkan *test case* untuk menguji dan mengevaluasi hasilnya. *Test case* menjamin bahwa semua jalur independen didalam modul telah dieksekusi sedikitnya satu kali, melaksanakan semua keputusan logis pada sisi benar dan yang salah, melaksanakan semua *loop* pada batas dan dalam batas-batas operasional mereka, dan melakukan struktur data internal untuk memastikan kesahihannya.

1. Notasi Grafik Alir

Mempertimbangkan metode jalur dasar, sebuah notasi sederhana untuk representasi aliran kontrol yang disebut grafik alir (atau grafik program). Grafik alir (*flow graph*) menggambarkan arus kontrol logis .



Gambar 2.4 Notasi Grafik Alir

Lingkaran (*node*), menggambarkan satu/lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu node. Tanda panah (*edge*), menggambarkan aliran kontrol. Setiap node harus mempunyai tujuan node. Region adalah daerah yang dibatasi oleh edge dan node. Termasuk daerah diluar grafik alir.

2. Cyclomatic Complexity

Cyclomatic complexity adalah metrik *software* yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam konteks metode uji coba basis path, nilai yang dihitung untuk cyclomatic complexity menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali. Jalur independent adalah jalur yang melintasi atau melalui program dimanasekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru. *Cyclomatic complexity* digunakan untuk mencari jumlah path dalam satu *flowgraph*. Dapat dipergunakan rumusan sebagai berikut :

a. Jumlah region grafik alir sesuai dengan cyclomatic complexity.
b. Cyclomatix complexity V(G) untuk grafik alir dihitung dengan rumus:

$$V(G) = E - N + 2 \qquad (2.2)$$

Dimana:

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

3. Cyclomatix complexity V(G) juga dapat dihitung dengan rumus:

$$V(G) = P + 1 \quad (2.3)$$

Dimana P = jumlah predicate node pada grafik alir

3. Graph Matriks

Prosedur ntuk menurunkan grafik alir dan bahkan menemukan satu set jalur dasar selalu selaras dengan mekanisasi. Sebuah struktur data yang disebut matriks grafik dapat sangan berguna untuk mengembangkan perangkat lunak yang membantu pengujian jalur dasar.

Matriks grafik adalah matriks persegi yang ukurannya (jumlah, baris, dan kolom) sama dengan jumlah node pada grafik alir. Setiap baris dan kolom terkait dengan node yang diidentifikasi, dan masukan matriks terkait dengan node yang diidentifikasi, dan masukan matriks terkait dengan koneksi-koneksi (edge) yang berada di antara node-node. Sampai saat ii matriks grafik tidak lebih dari sebuah representasi tabular dari grafik alir. Namun, dengan menambahkan bobot tautan (*link weight)* ke setiap entri matriks, matriks grafik dapat menjadi alat yang amat berguna untuk mengevaluasi struktur pengendali program selama pengujian dilakukan. Bobot tautan memberikan informasi tambahan tentang pengendalian aliran. Dalam bentuknya yang paling sederhana, bobot tautan bernilai 1 (ada sambungan) atau 0 (tidak ada sambungan). Akan tetapi, bobot tautan dapat diberi properti lain yang lebih menarik :

a. Kemungkinan bahwa sebuah tautan (koneksi) akan dieksekusi.
b. Waktu pengolahan yang dibutuhkan selama perjalanan dari sebua tautan.
c. Memori yang diperlukan selama perjalanan dari sebuah tautan.
d. Sumber daya yang diperlukan selama perjalanan dari sebuah tautan.

Beizer menyediakan perlakuan menyeluruh dari algoritma matematika tambahan yang dapat diterapkan pada matriks graph. Analisis yang diperlukan untuk merancang kasus uji dapat seluruhnya atau sebagian diotomatisasi dengan menggunakan teknik-teknik tersebut [11].

2.4.2 Black Box Testing (Pengujian Kotak Hitam)

Pengujian *black box* disebut juga dengan pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black box* bukan teknik alternatif untuk *white box*. Sebaliknya, *black box* merupakan pendekatan pelengkap yang mungkin dilakukan untuk mengungkap kelas kesalahan yang berbeda dari yang diungkap oleh metode *white box*. Pengujian *black box* berupaya untuk menemukan kesalahan dalam kategori berikut :

a. Fungsi yang salah atau hilang
b. Kesalahan antarmuka
c. Kesalahan dalam struktur data atau akses basis data eksternal
d. Kesalahan perilaku atau kinerja
e. Kesalahan inisialisasi dan penghentian.

Tidak seperti pengujian *white box*, yang dilakukan pada awal proses pengujian, pengujian *black box* cenderung diterapkan selama tahap-tahap pengujian selanjutnya. Pengujian black box dirancang untuk menjawab pertanyaan berikut :

a. Bagaimana validitas fungsional diuji?
b. Bagaimana perilaku dan kinerja sistem diuji?
c. Kelas-kelas masukan apakah yang akan membentuk *test case* yang baik?
d. Apakah sistem sangat sensitif terhadap nilai masukan tertentu?
e. Bagaimana batas-batas kelas data diisolasi?
f. Beberapa kecepatan dan volume data yang dapat ditolerir oleh sistem?
g. Apa pengaruh kombinasi spesifik data pada operasi sistem?