BAB III

METODOLOGI PENELITIAN

3.1 Alat dan Bahan Penelitian

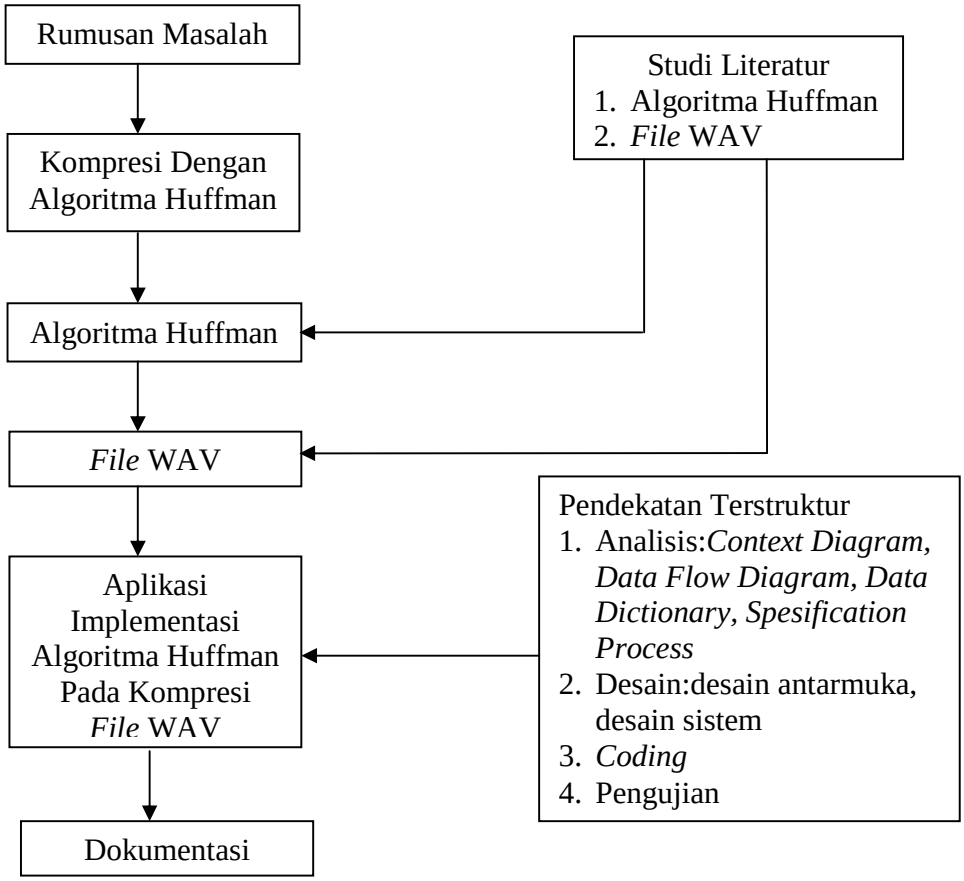
3.1.1 Alat Penelitian

1. Spesifikasi laptop yang digunakan dalam penelitian ini adalah sebagai berikut:
   - Processor AMD Turion 64 X2 Dual Core 1,66 Ghz
   - VGA NVDIA GeForce Go 6150 256 MB
   - Speaker Altec Lansing with Conexant HD Audio Output driver
   - RAM 2 GB
   - Hardisk 100 GB dengan *freespace* 18 GB
   - Layar Monitor 14 inch dengan kemampuan resolusi 1280 x 800 pixel, 32 bit color
   - Mouse dan Keyboard
2. Sistem operasi menggunakan Microsoft Windows XP Professional Version 2002 Service Pack 2 atau versi lain yang dapat mendukung Microsoft Visual Basic 6.0
3. Perangkat lunak untuk perancangan sistem:
   - Microsoft Visual Basic 6.0

3.1.2 Bahan Penelitian

Bahan dari penelitian ini merupakan *file-file* WAV yang terdapat pada sistem operasi Windows. *File* WAV yang digunakan merupakan bentuk format *file* suara tanpa kompresi. Format ini menyimpan semua detil suara yang biasanya berupa dua kanal suara, 44100hz *sampling rate*, 16 bit setiap *sample*. WAV biasanya menyimpan format PCM (*Pulse Code Modulation*) yang juga merupakan format standar audio untuk CD. Tetapi audio CD tidak memakai format WAV melainkan memakai *red book* audio format. Tetapi karena memakai format PCM maka data yang disimpan sama hanya berbeda pada headernya. Karena tidak di kompresi maka absennya suara tidak menjadikan ukuran file berubah tidak seperti format *lossy*. Tetapi WAV masih sering digunakan sebagai *master record* karena kualitasnya yang maksimal. (Adhinoto, 2010, *http://blog.ub.ac.id/teguh0610630106/2010/04/06/format-file-audio-wav/*)

3.2 Desain Penelitian



Gambar 3.1 Desain Penelitian

1. Rumusan Masalah. Rumusan masalah merupakan dasar pemikiran dan merupakan acuan dalam penelitian ini. Dalam penelitian ini, permasalahan yang akan di analisis adalah mengenai kompresi dengan algoritma Huffman. Untuk lebih jelas mengenai rumusan masalah dari penelitian ini dapat dilihat pada subbab 1.2 rumusan masalah.

2. Studi Literatur. Proses studi literatur dalam penelitian ini dilakukan dengan mempelajari literatur-literatur yang meliputi konsep Algoritma Huffman, teori kompresi, struktur *file* WAV, serta teori-teori lain yang mendukung pengembangan perangkat lunak ini.
3. Algoritma Huffman adalah algoritma yang akan diimplementasikan dalam perangkat lunak yang akan dikembangkan.
4. *File* WAV adalah objek yang akan dikompresi dengan menggunakan algoritma Huffman. Bagian *file* yang akan dikompresi adalah *chunk data*, yaitu *byte* ke 45 sampai *byte* akhir *file*.
5. Metode pendekatan yang digunakan dalam penelitian ini menggunakan pendekatan Terstruktur dengan model proses *prototype*.
6. Sistem Implementasi Algoritma Huffman Pada Kompresi *File* WAV atau disebut WAVCOMP merupakan nama perangkat lunak yang dikembangkan
7. Dokumentasi berupa dokumen teknis perangkat lunak, paper dan dokumen skripsi sebagai hasil dari penelitian.

3.3 Metode Penelitian

Dalam skripsi ini metode yang digunakan dalam penelitian meliputi metode pengumpulan data dan pengembangan perangkat lunak.

3.3.1 Metode Pengumpulan Data

Metode pengumpulan data dalam skripsi ini adalah studi literatur. Studi literatur dilakukan dengan mengumpulkan dan mempelajari literatur yang berkaitan dengan skripsi ini, seperti teori dan konsep kompresi, Algoritma Huffman dan pembahasan mengenai seluk-beluk/struktur *file* WAV melalui literatur-literatur seperti buku (*textbook*), paper, dan sumber ilmiah lain seperti situs intenet ataupun artikel dokumen teks yang berhubungan.

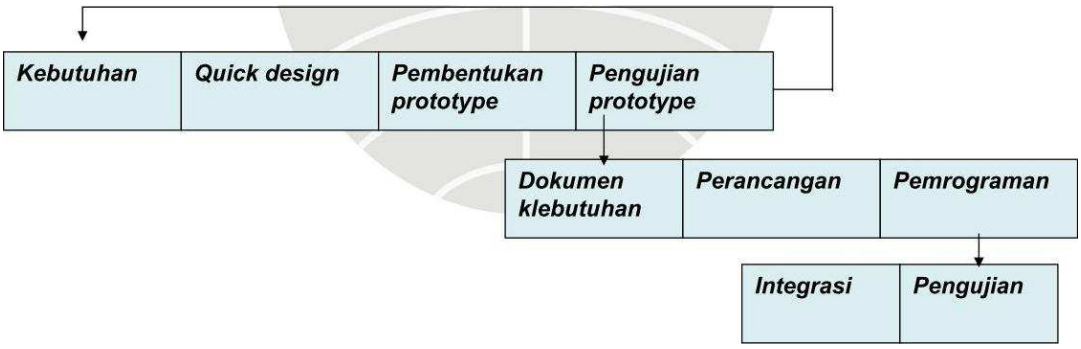
3.3.2 Metode Pengembangan Perangkat Lunak

Dalam pengembangan perangkat lunak, penulis menggunakan model *Prototype*. Pendekatan *Prototyping* adalah proses iterative yang melibatkan hubunan kerja yang dekat antara perancang dan pengguna.

Pressman (2001) menyatakan bahwa seringkali seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak mengidentifikasi kebutuhan *input*, pemrosesan, ataupun *output* detail. Pada kasus yang lain, pengembang mungkin tidak memiliki kepastian terhadap efisiensi algoritme, kemampuan penyesuaian dari sistem operasi, atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dan mesin.

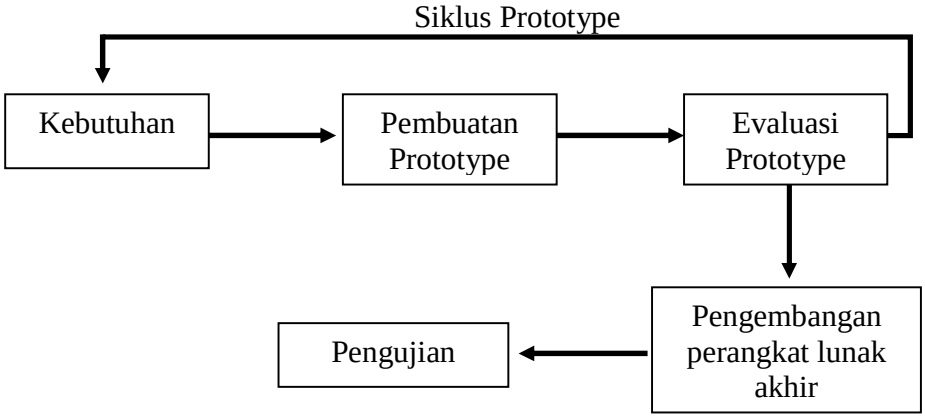
Pendekatan *Prototype* melewati tiga proses, yaitu analisis kebutuhan, pembuatan *Prototype*, dan evaluasi *Prototype*. Perulangan ketiga proses ini terus

berlangsung hingga semua kebutuhan terpenuhi. *prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk memahami kebutuhan klien lebih baik. *Prototype* yang dibuat dapat dimanfaatkan kembali untuk membangun *software* lebih cepat.



Gambar 3.2 Siklus Prototype (Wahyudin, Asep, 2007, h. 7)

Berdasarkan gambar siklus *Prototype* diatas, berikut ini adalah gambar siklus *Prototype* yang sudah disesuaikan dengan penelitian penulis :



Gambar 3.3 Model Pengembangan Perangkat Lunak

Adapun aktivitas-aktivitas yang dilalui sebagai berikut:

1. Analisis Kebutuhan

   Pada tahap awal dilakukan analisis kebutuhan, proses ini dilakukan untuk mengetahui informasi, model, dan spesifikasi dari sistem yang dibutuhkan.

2. Pembuatan Prototype

   Pada tahap ini, akan dilakukan pembuatan prototype sesuai dengan kebutuhan.

3. Evaluasi Prototype

   Tahap dimana prototype dievaluasi apakah sudah cocok atau belum dengan kebutuhan.

4. Pengembangan perangkat lunak akhir

   Melakukan pembuatan perangkat lunak yang telah cocok sesuai dengan kebutuhan sekaligus melakukan penyelesaian pengembangan perangkat lunak.

5. Pengujian (*Testing*)

   Tahapan selanjutnya adalah proses pengujian perangkat lunak, proses pengujian ini dilakukan untuk memastikan perangkat lunak yang telah dibuat telah sesuai dengan kebutuhan.

McLeod dan Schell (2001) mengemukakan bahwa alasan-alasan pemakai maupun spesialis informasi menyukai model *prototype* adalah:

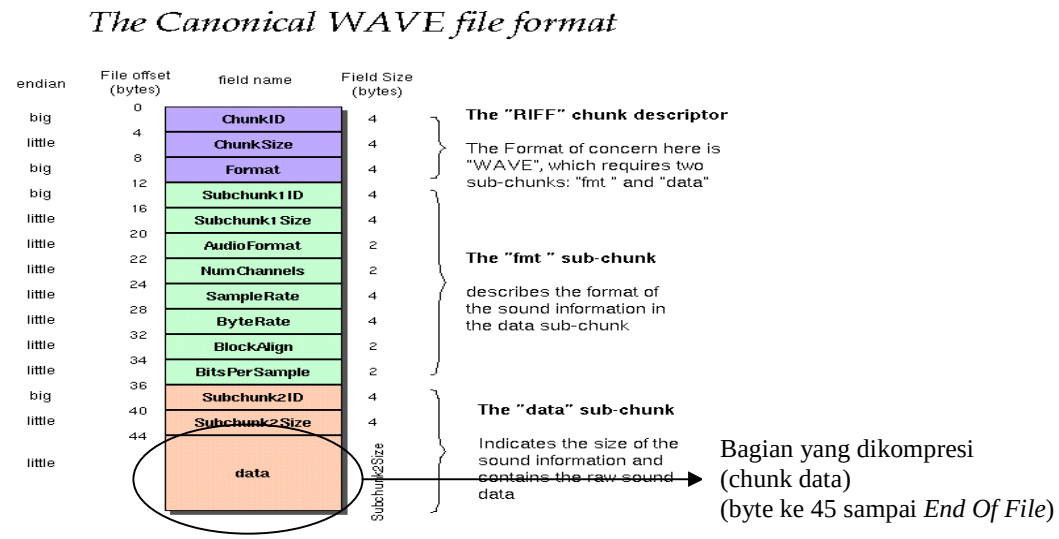
1. Komunikasi antara analis sistem dan pemakai membaik;
2. Analis dapat bekerja dengan lebih baik dalam menemukan kebutuhan pemakai;
3. Pemakai berperan lebih aktif dalam pengembangan sistem;
4. Spesialis informasi dan pemakai menghabiskan lebih sedikit waktu dan usaha dalam mengembangkan sistem;
5. Implementasi menjadi lebih mudah karena pemakai mengetahui sistem yang diharapkan.

3.4 Model Yang Dikembangkan

Model yang dikembangkan dalam penelitian ini adalah kompresi dan dekompresi algoritma Huffman pada *file* WAV.

3.4.1 File WAV

File WAV merupakan objek yang akan dikompresi. Seperti yang sudah dibahas, bagian *file* WAV yang akan dikompresi hanya bagian *chunk data*, sedangkan bagian lain tidak akan dikompresi. *Chunk data file* WAV terdapat pada *byte* ke 45 hingga akhir *file*.



Gambar 3.4 Bagian File WAV Yang Dikompresi

3.4.2 Algoritma Huffman

Algoritma Huffman sebenarnya merupakan algoritma kompresi yang dapat diterapkan pada semua jenis baik untuk *file* biner maupun *file* teks. Algoritma ini efektif dengan rasio kompresi yang rendah jika terdapat banyak *redundancy* data atau perulangan data yang sama pada *file*.

Pada program ini hanya akan dibuat kompresi dan dekompresi khusus hanya pada *file* audio berjenis WAV dan mempunyai *audio format* berjenis PCM (*Pulse Code Modulation*) dan hanya mendukung jumlah kanal maksimum 2 buah kanal (*mono* dan *stereo*). Untuk jenis WAV lainnya tidak dapat dilakukan proses kompresi.

3.4.3 Proses Kompresi Algoritma Huffman

Berikut ini langkah-langkah proses kompresi *file* WAV dengan algoritma Huffman :

1. Baca *file* dari *byte* awal hingga *byte* akhir data.
2. Ambil informasi *audio format file*, yaitu pada *byte* ke 21 untuk dicek apakah *file* valid untuk dikompresi.
3. Apabila *file* valid, ambil *chunk data file*, yaitu *byte* ke 45 hingga *byte* akhir data karena bagian ini yang akan dikompresi. Jika tidak valid, *file* tidak akan diproses.
4. Urutkan *byte-byte* data berdasarkan frekuensi kemunculan dari yang paling banyak hingga paling sedikit muncul.
5. Mulai pembentukan pohon Huffman. Ambil dua *byte* data yang paling sedikit muncul untuk digabungkan menjadi satu akar pohon dengan dua *leaf*/daun, urutkan kembali berdasarkan frekuensi kemunculan.
6. Ulangi langkah 4 dan 5 sampai akhir *byte file* sehingga membentuk 1 akar pohon tunggal.
7. Beri kode *bit* ‘0’ untuk setiap *leaf*/daun sisi kiri dan kode *bit* ‘1’ untuk setiap *leaf*/daun sisi kanan.
8. Buat kode *bit* untuk setiap *byte* data dengan menggunakan pohon Huffman, simpan informasi kode *bit* dari setiap *byte* beserta frekuensi kemunculan *byte* tersebut dalam tabel Huffman.
9. Karena 1 *byte*=8 *bit* maka kode *bit* yang terbentuk harus kelipatan 8, apabila tidak maka dilakukan penambahan ekstra kode *bit* ‘0’ pada akhir *file* sebanyak

yang dibutuhkan. Ini diperlukan untuk menulis *byte-byte* data *file* pengganti (*file* hasil kompresi).

10. Simpan informasi tabel Huffman dan penambahan ekstra kode *bit* tersebut pada header *chunk data file*. Ini diperlukan untuk proses dekompresi.
11. Ubah kode *bit* menjadi *byte-byte* data *file* pengganti (*byte file* hasil kompresi). Karena 1 *byte*=8 *bit* maka kelompokkan kode *bit* masing-masing 8 *bit*.
12. *Byte-byte* data *file* WAV berupa bilangan heksa. 1 bilangan heksa=4 *bit*, jadi 1 *byte* data *file*=2 bilangan heksa.
13. Ubah kode *bit* yang telah dikelompokkan tersebut menjadi menjadi *byte-byte* data *file* pengganti (*byte file* hasil kompresi) yang berupa bilangan heksa berdasarkan aturan pada nomor 12.
14. Ubah *audio format file* menjadi *audio format file* terkompresi.
15. Tulis kembali menjadi sebuah *file* WAV dari *byte-byte* data *file* pengganti (*byte file* hasil kompresi) tersebut dan header *chunk data file* yang telah berisi informasi tabel kode Huffman.

3.4.4 Proses Dekompresi Algoritma Huffman

Dekompresi dapat dilakukan dengan dua cara, yang pertama dengan menggunakan pohon Huffman dan yang kedua dengan menggunakan tabel kode Huffman. Pada penelitian ini, penulis menggunakan table kode Huffman.

Berikut ini langkah-langkah proses dekompresi *file* WAV dengan algoritma Huffman :

1. Baca *file* dari *byte* awal hingga *byte* akhir data.

2. Ambil informasi *audio format file*, yaitu pada *byte* ke 21 untuk dicek apakah *file* valid untuk didekompresi.
3. Apabila *file* valid, ambil *chunk data file*, yaitu *byte* ke 45 hingga *byte* akhir data karena bagian ini yang akan dikompresi. Jika tidak valid, *file* tidak akan diproses.
4. *Byte-byte* data *file* WAV berupa bilangan heksa. 1 bilangan heksa=4 *bit*, jadi 1 *byte* data *file*=2 bilangan heksa.
5. Ubah *byte-byte* data *file* terkompresi menjadi kode *bit* kembali, 1 *byte*=8 *bit*. Ubah berdasarkan aturan nomor 4. Periksa header *chunk data file*, apakah kode *bit* yang terbentuk telah mengalami penambahan ekstra *bit* pada proses kompresi. Apabila iya, kurangi kode *bit* yang terbentuk sebanyak penambahan yang dilakukan.
6. Setelah itu, ubah kode *bit* tersebut menjadi *byte-byte* data *file* asli (tidak terkompresi) berdasarkan informasi yang terdapat pada tabel Huffman.
7. Kode *bit* untuk setiap *byte* data tidak boleh menjadi awalan dari kode *bit* setiap *byte* data yang lain guna menghindari keraguan (ambiguitas) dalam proses dekompresi.
8. Baca *bit* pertama dari kode *bit*, lakukan pengecekan pada tabel Huffman apakah kode *bit* tersebut merupakan kode *bit* dari salah satu *byte* data.
9. Apabila iya, ubah kode *bit* tersebut menjadi *byte* data kembali. Apabila bukan, baca kembali *bit* selanjutnya hingga kode *bit* tersebut merupakan kode *bit* dari suatu *byte* data.
10. Ulangi langkah 8 dan 9 hingga akhir kode *bit*.

11. Ubah *audio format file* menjadi *audio format file* asli (tidak terkompresi). Hapus informasi tabel Huffman pada header *chunk data file*.
12. Tulis kembali menjadi sebuah *file* WAV dari *byte-byte* data *file* dan header *file*.

3.5 Fokus Penelitian

Fokus dari penelitian ini adalah sebagai berikut :

1. Mengembangkan aplikasi atau *prototype* kompresi dan dekompresi *file* WAV dengan agoritma Huffman.
2. Meneliti besarnya pengurangan ukuran *file* WAV setelah dikompresi.
3. Meneliti hasil *file* yang didekompresi, apakah kembali seperti semula atau tidak.