BAB II

LANDASAN TEORI

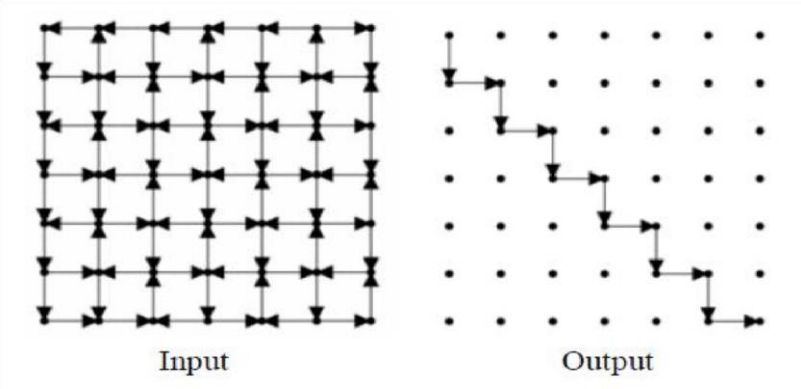
2.1. Pencarian Rute Terpendek

Lintasan terpendek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf adalah sekumpulan titik di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (*edge*) . Sebuah graf dibentuk dari kumpulan titik yang dihubungkan dengan garis-garis. Ada beberapa istilah yang berhubungan dengan graph, antara lain (Taurus, 2012):

1. Titik – titik tersebut disebut *vertex*.
2. Garis – garis yang menghubungkan antar *vertex* disebut *edge*.
3. *Adjacent* artinya bertetangga. Maksudnya jika ada dua *vertex* disebut *adjacent*, jika mempunyai *edge* yang sama.
4. Adalah bobot yang biasanya terdapat pada *edge* yang merepresentasikan jarak dari *vertex-vertex* yang dihubungkan oleh *edge* tersebut.
5. *Path* adalah lintasan yang melalui *edge* dan *vertex* dalam graf.
6. *Cycle* adalah lintasan yang dimulai dan berakhir pada *vertex* yang sama.
7. *Direct* pada *directed graph* adalah graf dimana *edge-edge*nya mempunyai suatu arah.

Graf yang digunakan adalah graf-graf yang berbobot, yaitu graf yang setiap sisinya diberikan suatu nilai. Ada beberapa macam persoalan lintasan terpendek, antara lain (Arsa, 2012):

1. Lintasan terpendek antara dua buah simpul tertentu. (*a pair shortest path*).
2. Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*singlesource shortest path*).
4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).



Gambar 2.1. Contoh Pencarian Rute Terpendek

2.2. Pengenalan Algoritma Pencarian Rute Terpendek

Algoritma adalah kumpulan perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis (logika) untuk penyelesaian suatu masalah. Namun algoritma pencarian rute tujuannya adalah algoritma yang menentukan bagaimana memilih rute optimal antara awal dan tujuan dengan memperhitungkan waktu

kalkulasi terpendek. Ada beberapa Algoritma yang sudah dikembangkan, antara lain Algoritma *Dijkstra,* Algoritma *Floyd-Warshall,* Algoritma *Bellman-Ford*, Algoritma *Ant*, Algoritma *A**, dan lain-lain. Dimana inti logika dari algoritma-algoritma tersebut adalah sama, yaitu menentukan jarak terpendek dari setiap titik yang telah dibangun (Siswanto, 2010).

Dari penelitian diatas dapat dilihat bahwa, algoritma-algoritma ini memiliki kelebihan dan kekurangannya masing-masing dalam menyelesaikan persoalan lintasan terpendek, karena masing-masing algoritma memiliki spesifikasi penyelesaian masalah, kompleksitas, waktu penyelesaian, serta jenis masalah yang berbeda, dan yang menjadi dasar serta alasan mengapa Algoritma *Dijkstra* yang digunakan yaitu untuk membantu memecahkan masalah pada tugas akhir ini, karena Algoritma *Diijkstra* lebih memandang solusi akhir yang akan diperoleh sebagai sesuatu keputusan yang saling terkait. Artinya solusisolusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya

2.2.1. Algoritma *Dijkstra*

Algoritma *Dijkstra* adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan juga sekaligus untuk semua pasangan titik. Ketiadaan garis yang menghubungkan sebuah pasangan dilambangkan dengan tak-hingga. Dalam

pengertian lain Algoritma *Dijkstra* adalah Algoritma yang akan memilih satu jalur terpendek dan teraman dari beberapa alternatif jalur yang telah dihasilkan dari proses kalkulasi.

2.2.1.1. Deskripsi Algoritma *Dijkstra*

Dasar algoritma *Dijkstra* adalah sebagai berikut:

a. Asumsikan semua simpul graf berarah G adalah V = {1, 2, 3, 4, ..., n}, perhatikan subset {1, 2, 3, ..., k}.

b. Untuk setiap pasangan simpul i, j pada V, perhatikan semua lintasan dari i ke j dimana semua simpul pertengahan diambil dari {1, 2, ..., k}, dan p adalah lintasan berbobot minimum diantara semuanya.

c. Algoritma ini mengeksploitasi relasi antara lintasan p dan lintasan terpendek dari i ke j dengan semua simpul pertengahan berada pada himpunan {1, 2, ..., k−1}.

d. Relasi tersebut bergantung pada apakah k adalah simpul pertengahan pada lintasan p. Berikut adalah *pseudocode* Algoritma *Dijkstral*:

```
function fw (int[1 ..n,1..n] graph)  {

//  Inisialisasi  var  int[1
..n,1..n] jarak := graph var
int[1 ..n,1..n] sebelum for
i from 1  to n for j from 1
```

```
to n if jarak[i,j] < Tak-
hingga sebelum[i,j] := i

// Perulangan utama pada algoritma for k
from 1 to n for i from 1 to n for j
from 1 to n if jarak[i,j] > jarak[i,k]
+ jarak[k,j] jarak[i,j] = jarak[i,k] +
jarak[k,j] sebelum[i,j] = sebelum[k,j]

return jarak

}
```

Algoritma 2.2. *Pseudocode* Algoritma *Dijkstra*

2.2.1.2. Karakteristik Algoritma *Dijkstra*

Beberapa karakteristik yang dimiliki oleh algoritma dijkstra antara lain (Novandi, 2010):

1. Persoalan dibagi atas beberapa tahap.
2. Ketika masuk ke suatu tahap, hasil keputusan akan manjadi simpul baru.
3. Bobot pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.
4. Bobot yang ada pada suatu tahap tergantung dari bobot tahapan yang telah dilewati dan bobot pada tahap itu sendiri.

5. Keputusan terbaik pada suatu tahap berkaitan terhadap keputusan pada tahap sebelumnya.

6. Terdapat hubungan yang menyatakan bahwa keputusan terbaik dalam setiap status pada tahap **k** akan memberikan keputusan terbaik untuk setiap status pada tahap **k + 1**.

7. Prinsip optimalitas berlaku pada algoritma ini.

2.2.1.3. Pemecahan Masalah Algoritma *Dijkstra*

Algoritma *dijkstra* membandingkan semua kemungkinan lintasan pada graf untuk setiap garis dari semua titik. Misalkan terdapat suatu graf G dengan simpul-simpul V yang masing-masing bernomor 1 sampai n (sebanyak n buah). Misalkan pula terdapat suatu fungsi *shortestPath* (i, j, k) yang mengembalikan kemungkinan jalur terpendek dari i ke j dengan hanya memanfaatkan simpul 1 sampai k sebagai titik perantara. Tujuan akhir penggunaan fungsi ini adalah untuk mencari jalur terpendek dari setiap simpul i ke simpul j dengan perantara simpul 1 sampai k+1.

Ada dua kemungkinan yang terjadi:

1. Jalur terpendek yang sebenarnya hanya berasal dari simpul-simpul yang berada antara 1 hingga k.

2. Ada sebagian jalur yang berasal dari simpul-simpul i sampai k+1, dan juga dari k+1 hingga j

Perlu diketahui bahwa jalur terpendek dari i ke j yang hanya melewati simpul 1 sampai k telah didefinisikan pada fungsi *shortestPath*(i, j, k) dan telah jelas bahwa jika ada solusi dari i sampai k+1 hingga j, maka panjang dari solusi tadi adalah jumlah dari jalur terpendek dari i sampai k+1 (yang melewati simpulsimpul 1 sampai k), dan jalur terpendek dari k+1 sampai j (juga menggunakan simpul-simpul dari 1 sampai k). Maka dari itu, rumus untuk fungsi *shortestPath*(i, j, k) bisa ditulis sebagai suatu notasi sebagai berikut.:

```
Basis-0

shortestPath (i, j, 0) = edgeCost

( i, j);


Rekurens

   shortestPath (i, j, k) = min

(shortestPath (i, j, k-1)  ,

shortestPath (i, k, k-1) + shortestPath (k, j, k-1)) ;
```

Algoritma 2.3. Pemecahan Solusi Rute Terpendek

Rumus ini adalah inti dari Algoritma *dijkstra*, algoritma ini bekerja dengan menghitung *shortestPath*(i,j,1) untuk semua pasangan (i,j), kemudian hasil tersebut akan digunakan untuk menghitung *shortestPath*(i,j, 2) untuk semua pasangan (i,j), dan seterusnya. Proses ini akan terus berlangsung hingga k = n dan kita telah

menemukan jalur terpendek untuk semua pasangan (i,j) menggunakan simpul-simpul perantara.

2.3. Analisa dan Perancangan

Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah untuk mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata (Widodo, 2011).

Perusahaan *software*, *Rational Software*, telah membentuk konsorsium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysis Design* (*OOAD).*

2.3.1. *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Widodo, 2011).

Untuk merancang sebuah model, UML memiliki beberapa diagram antara lain : *use case diagram, class diagram, statechart diagram, activity diagram,*

sequence diagram, collaboration diagram, component diagram, deployment diagram.

2.3.2. *Use Case Diagram*

Use case diagram merupakan sebuah gambaran fungsionalitas sebuah sistem. Sebuah *use case* merepresentasikan interaksi antara aktor dengan sistem.

Use case sangat menentukan karakteristik sistem yang sedang dibuat. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu (Widodo, 2011).

Dalam sebuah sistem *use case diagram* akan sangat membantu dalam hal menyusun *requirement*, mengkomunikasikan rancangan dengan klien dan merancang *test case* untuk semua fitur yang ada pada sistem.

2.3.3. *Class Diagram*

Class merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) (Widodo, 2011).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu nama, *stereotype*, atribut dan metoda.

2.3.4. *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* biasa 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 (Dharwiyanti, 2006).

2.4. *Rational Unified Process* (RUP)

Untuk pengembangan aplikasi pencarian rute terpendek lokasi fasilitas umum berbasis Android pada tugas akhir ini menggunakan metode pengembangan perangkat lunak *Rational Unified Process* (RUP).

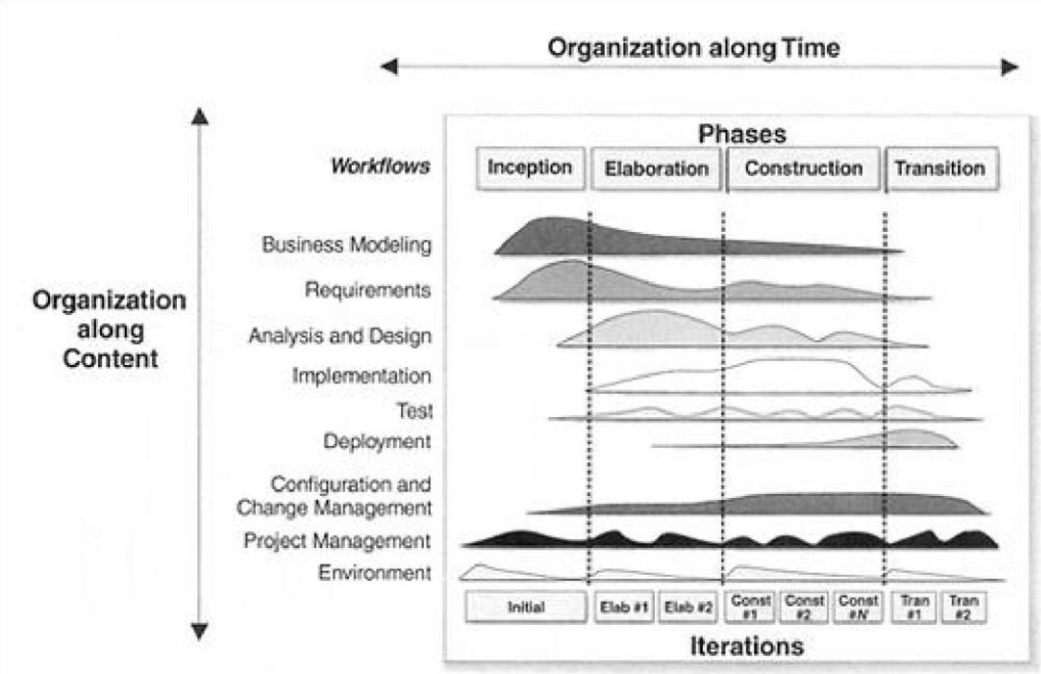
2.4.1. Pengertian RUP

Rational Unified Process adalah sebuah Proses Rekayasa Perangkat Lunak. RUP menyediakan pendekatan disiplin untuk memberikan tugas dan tanggung jawab dalam organisasi pengembang perangkat lunak. Tujuannya untuk memastikan perangkat lunak yang berkualitas tinggi dan sesuai kebutuhan penggunanya dalam anggaran dan jadwal yang dapat diprediksi (Kruchten, 2000).

RUP mengarahkan kita terhadap pengembangan perangkat lunak secara praktis dan efektif. Terdapat 6 *best practice* atau disebut juga *basic principle* dalam metode RUP, antara lain (Kruchten, 2000):

1. *Develop software iteratively*, bertujuan untuk mengurangi resiko pada awal proyek.
2. *Manage requirements*, bertujuan untuk mengatur kebutuhan yang diperlukan selama proyek.
3. *Use component-based architectures* untuk membangun komponen arsitektur sebuah proyek.
4. *Visually model software*, bertujuan untuk merancang sebuah model visual perangkat lunak, untuk mendapatkan struktur dan perilaku dari aritektur perangkat lunak.
5. *Continuously verify software quality*.
6. *Control changes to software.* kemampuan untuk mengatur serta mengubah perangkat lunak saat dibutuhkan.

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui Gambar 2.4 dibawah dapat dilihat bahwa RUP memiliki 2 dimensi, yaitu:



Gambar 2.4. Struktur Proses 2 Dimensi RUP

Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu atau beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.

Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspekaspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing*, *what*, *how* dan *when*. Dimensi ini terdiri atas *Business Modeling*, *Requirement*,

Analysis and Design, Implementation, Test, Deployment, Configuration dan *Change Manegement, Project Management, Environtment.*

2.4.2. Fase RUP

Fase-fase pada RUP berdasarkan waktu pengerjaan proyek dapat dibagi menjadi 4 fase, yaitu *Inception*, *Elaboration*, *Construction* dan *Transition* (Rational Team, 2001).

1. Fase *Inception*

   Fase *inception* merupakan fase untuk mengidentifikasi masalah, untuk itu diperlukan juga idetifikasi entitas dari luar yang berhubungan dengan sistem. Pada fase ini melibatkan semua identifikasi *use case* dan gambaranya. Selain itu juga termasuk kriteria keberhasilan proyek, perkiraan resiko, perkiraan terhadap *resource* yang dibutuhkan dan merencanakan penjadwalan

   milestone. Hasil yang diperoleh pada fase ini adalah :

   a. Dokumen visi (visi dari kebutuhan projek, kata kunci, batasan utama).

   b. Inisialisasi model *use-case* (10%-20% selesai).

   c. Daftar kata.

   d. *Business case*, termasuk didalamnya konteks bisnis, kriteria sukses, pengenalan pasar dan proyeksi keuangan.

   e. Inisialisasi penialaian resiko.

f. Rencana proyek dan menunjukan fase serta iterasi.

g. Model bisnis jika diperlukan

Kriteria evaluasi untuk fase *Inception* adalah :

a. Menyesuaikan *stakeholder* dengan *scope definition* dan perkiraan biaya atau perkiraan jadwal.

b. Pemahaman terhadap *use-case* utama.

c. Kredibilitas dari perkiraan biaya, jadwal, prioritas, resiko dan proses pengembangan.

d. Pemahaman terhadap *prototype*.

2. Fase *Elaboration*

Tujuan dari fase *elaboration* (pengembangan) adalah menganalisa area permasalahan, mengembangkan rencana proyek, dan menghilangkan unsurunsur yang memiliki resiko besar terhadap proyek. Adapun hasil dari fase *elaboration* adalah:

a. *Use case* model, seluruh use case dan aktor telah teridentifikasi.

b. *Requirement* tambahan yang mungkin tidak bersifat fungsional bagi proyek.

c. *Software Architecture Description* (Deskripsi Arsitektur Perangkat Lunak).

d. Prototipe dari arsitektur yang dapat dieksekusi.

e. Revisi daftar tingkat resiko dan revisi *business-case*.

f. Rencana pengembangan keseluruhan proyek.

g. Persiapan dokumen panduan bagi pengguna (*user manual*).

Kriteria utama dalam fase elaboration melibatkan pertanyaan berikut :

a. Apakah produk sudah stabil ?

b. Apakah rancangan arsitekturalnya sudah stabil ?

c. Apakah saat demo prototipe, unsur yang memilki resiko telah bisa di atur ?

d. Apakah rencana kontruksi telah detail dan akurat ?

e. Apakah *stakeholder* bersedia dan menyepakati visi dari

pengembangan proyek tersebut?

f. Apakah pembelanjaan *actual-resource* terhadap rencana

pembelanjaan dapat diterima?

3. Fase *Contruction*

Selama fase kontruksi, semua komponen dan fitur yang dikembangkan terintergrasi ke dalam produk dan secara menyeluruh semua fitur telah diuji.

Di lain sisi, proses konstruksi adalah sebuah proses *manufacturing*, dimana terdapat penekanan dalam mengelola *resource* dan mengatur operasi untuk mengoptimalkan jadwal dan kualitas. Pada tahap ini pola pikir (*mindset*) mengalami perubahan dari pengembangan *intellectual property* pada fase *Inception* dan *Elaboration*, menjadi pengembangan *deployable product*. Kriteria evaluasi terhadap fase *Construction* ini adalah :

a. Apakah peluncuran produk cukup baik dan dapat diterima di komunitas pengguna?

b. Apakah semua *stakeholder* siap untuk beralih ke komunitas pengguna?

c. Apakah pembelanjaan *actual-resource* terhadap rencana pembelanjaan masih tetap diterima?

4. Fase *Transition*

Tujuan dari fase ini adalah untuk transisi dari produk perangkat lunak ke pengguna akhir. Apabila produk telah di luncurkan kepada pengguna, maka isu-isu akan muncul dari pengguna. Nantinya isu ini akan digunakan untuk tahap perbaikan terhadap produk. Kriteria evaluasi untuk fase *Transition* adalah :

a. Apakah pengguna merasa puas?

b. Apakah pembelanjaan *actual-resource* terhadap rencana pembelanjaan masih tetap diterima?

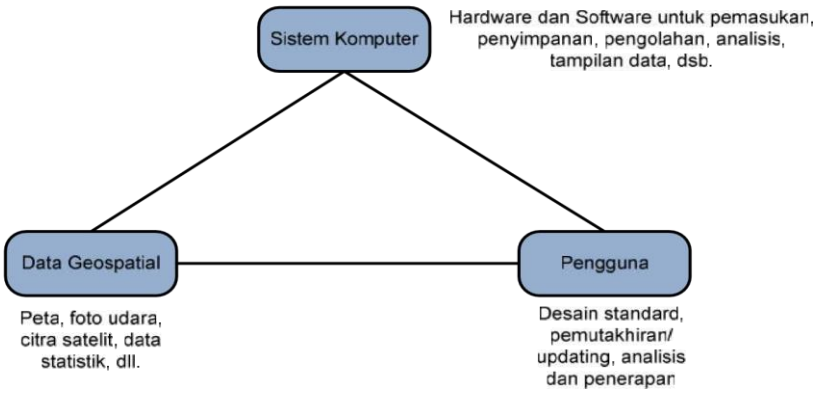
2.5. Sistem Informasi Geografis (SIG)

2.5.1. Pengertian Sistem Informasi Geografis (SIG)

Sistem Informasi Geografis (SIG) adalah sistem informasi yang digunakan untuk memasukkan, menyimpan, memanggil, mengolah, menganalisis dan menghasilkan data bereferensi geografis (permukaan bumi) atau data geospasial, untuk mendukung pengambilan keputusan dalam perencanaan dan pengelolaan penggunaan lahan, sumber daya alam, lingkungan transportasi, fasilitas kota, dan pelayanan umum lainnya.

Sumber lainnya menyatakan Sistem Informasi Geografis itu adalah kumpulan yang terorganisasi dari perangkat keras komputer, perangkat lunak, data geografi dan personil yang dirancang secara efisien untuk memperoleh, menyimpan, meng-*update*, memanipulasi, menganalisis dan menampilkan semua bentuk informasi yang bereferensi geografis (Wiley, 1990).

Komponen dari SIG adalah sistem komputer yang terdiri atas perangkat keras (*hardware*) dan perangkat lunak (*software*) , data geospasial dan pengguna (*brainware*) seperti yang ditunjukkan pada Gambar 2.5 (Rizal, 2011).



Gambar 2.5. Komponen Sistem Informasi Geografis

Data yang diolah pada SIG adalah data geospasial (data spasial dan data non-spasial). Biasanya data non-spasial tidak digambarkan karena memang dalam SIG yang dipentingkan adalah tampilan data secara spasial.

Data spasial sendiri itu adalah data yang berhubungan dengan kondisi geografi misalnya sungai, wilayah administrasi, gedung, jalan raya, dan sebagainya. Biasanya data spasial bisa didapatkan dari peta, foto udara, citra satelit, data statistik dan lain-lain. Secara umum persepsi manusia mengenai bentuk representasi entitas spasial adalah konsep raster dan vektor. Sedangkan data non-spasial adalah selain data spasial yaitu data yang berupa text atau angka yang biasa disebut dengan atribut.

Data non-spasial ini akan menerangkan data spasial atau sebagai dasar untuk menggambarkan data spasial. Dari data non-spasial ini nantinya dapat dibentuk data spasial. Misalnya jika ingin menggambarkan peta penyebaran penduduk maka diperlukan data jumlah penduduk dari masing-masing daerah (data non-spasial), dari data tersebut nantinya akan dapat digambarkan pola penyebaran penduduk untuk masing-masing daerah (spasial).

2.5.2. Subsistem SIG

Sistem Informasi Geografis (SIG) dapat diuraikan menjadi beberapa subsistem sebagai berikut (Nirwan, 2011):

1. Data Input

   Subsistem ini bertugas untuk mengumpulkan dan mempersiakan data spasial dan atribut dari berbagai sumber.

2. Data Output

Subsistem ini menampilkan atau menghasilkan keluaran seluruh atau sebagian basis data dalam bentuk *softcopy* maupun bentuk *hardcopy*.

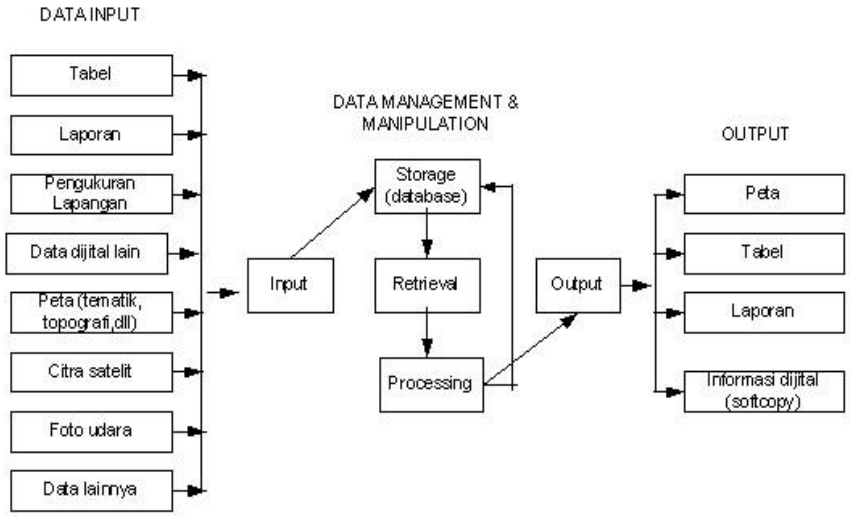
3. Data Management

Subsistem ini mengorganisasi baik data spasial maupun atribut kedalam sebuah basis data sedemikian rupa sehingga mudah dipanggil, di-*update*, dan di-*edit*.

4. Data Manipulation & Analysis

Subsistem ini menentukan informasi-informasi yang dapat dihasilkan oleh SIG. Subsistem ini juga melakukan manipulasi dan pemodelan data untuk menghasilkan informasi yang diharapkan.

Jika subsistem SIG tersebut diperjelas berdasarkan uraian jenis masukan, proses, dan jenis keluaran yang ada didalamnya, maka subsistem SIG dapat juga digambarkan seperti pada Gambar 2.5.



Gambar 2.6. Subsistem Sistem Informasi Geografis

2.5.3. Konsep Model Data Spasial SIG

Data spasial merupakan data yang paling penting dalam SIG. Data spasial ada 2 macam yaitu data raster dan data vektor (Prahasta, 2009):

a. Data Raster

Model data raster menampilkan, menempatkan dan menyimpan data spasial dengan menggunakan struktur matriks atau pixel-pixel yang membentuk grid. Konsep model data ini adalah dengan memberikan nilai yang berbeda untuk tiap-tiap pixel atau grid dari kondisi yang berbeda.

b. Data Vektor

Model data vektor yang menampilkan, menempatkan dan menyimpan data spasial seperti titik-titik, garis-garis, atau kurva atau poligon beserta atribut-atributnya. Bentuk dasar representasi data spasial didalam sistem model data vektor, didefenisikan oleh sistem koordinat kartesian dua dimensi (x,y).

2.5.4. *Universal Tranverse Mercator* (UTM)

UTM merupakan satuan koordinat berdasarkan satuan jarak dan berhubungan dengan proyeksi yang digunakan, yaitu konversi UTM. Proyeksi UTM adalah sistem proyeksi orthometrik dengan satuan panjang meter (m) berdasar *mercator* (bidang silinder) terhadap kedudukan bidang proyeksi *transversal* (melintang), menggunakan zona dengan interval 6º meridian yang dikenalkan oleh Mercator.

Koordinat UTM adalah koordinat ortometrik 2 dimensi, dengan titik acuan absis x dalam satuan E (*East*) awal 500.000 m dan ordinat y dalam satuan N (*North*) awal 10.000.000 m terletak di pusat proyeksi (perpotongan *Meridian Central* (MC) atau tengah zona dengan ekuator). Arah utara *grid* sejajar proyeksi zona MC**,** merupakan juring elipsoid dengan batasan 6º diawali di Bujur 180 º dengan arah Timur (zona 1) sampai dengan zona 60. Artinya berawal di Bujur 190º ketimur (Bujur Timur) melalui Bujur 0º di Greenwich (zona 30) berakhir di Bujur 180 Timur (zona 60) garis Bujur atau garis Meridian. Indonesia terletak pada zona 46 hingga zona 54. Kota Pekanbaru terletak pada zona 47 N (Prahasta, 2009)

Proyeksi potongan satu bidang dengan elipsoid melalui dua kutubnya yang merupakan garis di permukaan elipsoid bumi membujur dari Kutub Utara ke Kutub Selatan, dihitung dari Bujur 0º Greenwich 180º kearah Timur dan 180 º kearah Barat.

2.6. *Literatur Review*

1. Lestariningsih, Endang (2011), Merancang dan mengimplementasikan rute terpendek menggunakan Algoritma Genetik. Dalam penelitian ini dilakukan perancangan dan implementasi suatu perangkat lunak menggunkan borland delphi versi 5.0 sebagai alat bantu untuk menyelesaikan model jaringan rute terpendek menggunakan algoritma genetic

   Kekurangan adanya kepastian untuk menghasilkan solusi optimum global, karena sebagian besar dari algoritma ini berhubungan dengan bilangan random yang bersifat probabilistik.

2. Kartika, dkk (2011), Perencanaan Rute Perjalanan Di Jawa Timur Dengan Dukungan GIS, Penelitian ini mempunyai tujuan adalah merancang dan membuat suatu perangkat lunak yang dapat memberikan informasi geografi mengenai rute jalan terpendek antara kota yang satu dengan kota yang lainnya di Jawa Timur. Program ini dirancang tanpa menggunakan satelit namun hanya menggunakan database, sehingga penggunaannya lebih murah dibandingkan dengan menggunakan satelit.

   Kekurangan, kompleksitas waktu algoritma pada program, program menghasilkan waktu yang sangat lama sehingga perhitungannya akan semakin lama.

3. Prasojo, Ipin, dkk (2013). Adanya sumber informasi yang terpercaya tentang bencana alam sedikit banyak akan menimbulkan ketenangan di tengah-tengah masyarakat dalam menjalani hidupnya, sehingga dapat tercipta tingkat kewaspadaan untuk mengatasi bencana alam yang mungkin terjadi. Dalam tulisan ini, dijelaskan adanya sistem penyebaran luasan informasi bencana alam yang ditujukan untuk memberikan informasi kepada masyarakat tentang bencana alam dari sumber terpercaya secara cepat dan tepat sasaran.

   Kekurangan, sangan sulit untuk digunakan pada pemecahan masalah yang cepat, karena membutuhkan kumpulan banyak opsi terlebih dahulu sebelum dieksekusi. Hal ini akan membuat pertimbangan dalam memilih opsi akan menjadi lambat.

4. Anam (2016) menyatakan bahwa rute terbaik itu merupakan rute yang didapatkan dengan memperhatikan berbagai kondisi yang ada pada jalan yang dilewati seperti keadaan jalan rusak, kapasitas jalan, banyaknya kendaraan yang melewati jalan, dan sebagainya. Selain panjang jalan, faktor yang diperhatikan sebagai kondisi jalan pada penelitian Anam (2016) adalah kepadatan lalu lintas. Secara linguistik, kepadatan lalu lintas ini dapat dinyatakan dengan ’tidak padat’, ’sedang’, dan ’padat’. Logika fuzzy sering digunakan untuk memecahkan permasalahan yang terjadi di dunia nyata yang mempunyai nilai samar atau nilai diantara true dan false. Oleh karena itu, logika fuzzy digunakan untuk menentukan bobot pada graph berdasarkan kepadatan lalu lintas pada jalan.

   Kekurangan, data yang dihasilkan masih kurang valid, karena masih perkiraan kecepatan dalam memperoleh data masih dipengaruhi oleh nilai/jarak pada simpul.
5. Lestariningsih, Endan (2012) merancang dan mengimplementasikan rute terpendek menggunakan algoritma Floyd Warshall. Dalam penelitian ini dilakukan perancangan dan implementasi suatu perangkat lunak menggunakan borlan Delphi versi 5.0 sebagai alat bantu untuk menyelesaikan model jaringanrute terpendek menggunakan algoritma genetic. Rekayassa perangkat lunak meliputi perencanaan dan implementasi serta pengujian program. Analisa hasil program menunjukan bahwa penyelesaian masalah perancangan mendekati optimal apabila populasi yang digunakan semakin banyak.

Kekurangan, bahwa keputusan yang diambil pada tahapan hanya pada informasi yang terbatas, sehingga hanya nilai optimum yang di peroleh pada saat itu.