

BAB III

METODOLOGI PENELITIAN

3.1. Alat dan Bahan Penelitian

Alat dan Bahan Penelitian yang dimaksud disini adalah *Resource*, yaitu sumber daya yang mendukung pengerjaan sistem dari awal hingga selesai. Resource tersebut saya golongkan menjadi dua bagian besar, yaitu:

- Sumber Daya Manusia
- Sumber Daya Pendukung (Teknologi)

Sumber daya manusia yang dibutuhkan melibatkan keterbitatan keahlian di beberapa bidang, berikut adalah daftar sumber daya manusia beserta tugasnya, antara lain:

- **Data kolektor:** Data gedung, jarak antar gedung, foto gedung dan lokasi penelitian
- **Modeler objek 3D:** Membuat lokasi dari foto menjadi objek 3 dimensi
- **Programmer:** Membuat sistem interaktif yang dinamis dari objek 3D yang telah tersusun terbentuk rapi
- **Database administrator:** Merancang sistem database untuk informasi lokasi, baik informasi gedung maupun informasi daerah lainnya di sekitar lokasi
- **Interface Designer:** Membuat tampilan program yang nyaman dan memudahkan user dalam penggunaan program

Dibawah ini saya cantumkan 2 sumberdaya (selain SDM) sebagai pendukung pengerjaan dan penggunaan program sebagai hasil dari penelitian

Software: Sketchup 3D, Flex 3.0, Papervision 3D, AMFPHP, MySql Workbench.

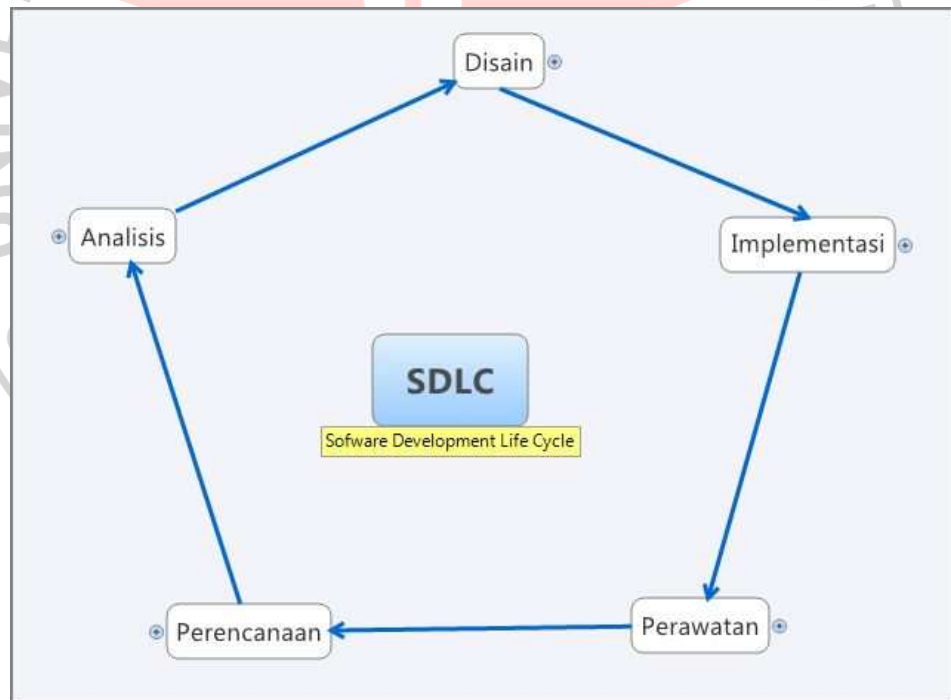
Hardware: Perangkat Komputer dengan Spesifikasi yang mumpuni untuk menjalankan Program yang berbasis sistem 3 Dimensi



3.2. *Disain Penelitian*

Penelitian dimulai dari pemilihan lokasi yang sesuai dengan kriteria perancangan sistem. Kriteria lokasi yang dikaji adalah lokasi yang memiliki sejumlah gedung yang banyak dan tersebar di beberapa lokasi. Untuk tahap awal dan berdasarkan kriteria diatas, saya mengambil lokasi di Universitas Pendidikan Indonesia. Untuk pengembangan selanjutnya, sistem dapat di terapkan dilokasi lain yang sesuai dengan kriteria yang ditentukan.

SDLC (Software Development Life Cycle)



Gambar 3.1 Siklus pengembangan software

Pengembangan perangkat lunak dapat dianggap sebagai lingkaran pemecahan masalah. Untuk menyelesaikan masalah besar, dipecah menjadi kecil terus-menerus sampai ke tingkat yang paling kecil, kemudian diselesaikan. Daur hidup perangkat lunak adalah model proses untuk rekayasa perangkat lunak yang dipilih berdasarkan sifat aplikasi dan proyeknya, metode dan tool yang digunakan, serta kontrol dan deliverable yang diinginkan.

Dalam perancangan perangkat lunak ini penulis menggunakan model Rapid Application Development (RAD). Model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Menekankan perkembangan komponen program yang bisa dipakai ulang (reusability) sehingga mendasari konsep Object-Oriented.

Fase pendekatan Rapid Application Development:

1. Business Modeling
2. Data Modeling
3. Proses Modeling
4. Application generation: RAD mengasumsikan pemakaian teknik 4G (generasi keempat). Selain menciptakan PL dengan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program atau menciptakan komponen yang bisa dipakai lagi.

5. Testing and Turn Over: Banyak komponen program yang telah diuji sebelumnya sehingga mengurangi keseluruhan waktu pengujian. Tapi komponen baru harus diuji dan semua interface harus dilatih secara penuh.

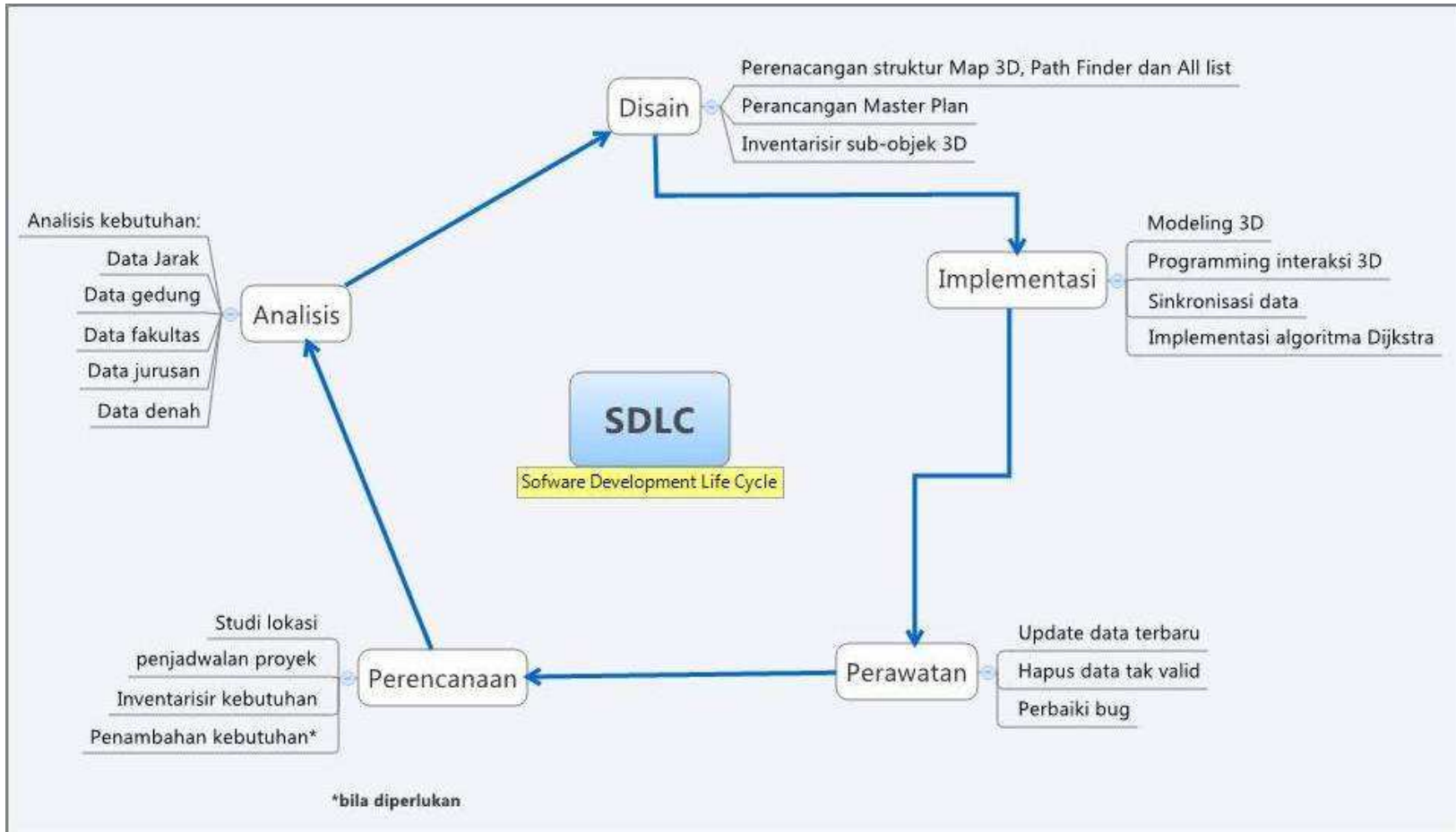
Model RAD adalah model proses pembangunan [perangkat lunak](#) yang tergolong dalam teknik incremental (bertingkat). RAD menekankan pada siklus pembangunan pendek/singkat/cepat. Waktu yang singkat adalah batasan yang penting untuk model ini. Model RAD mengadopsi model waterfall dan pembangunan dalam waktu singkat yang dicapai dengan menerapkan :

1. Component based construction (pemrograman berbasis komponen).
2. Penekanan pada penggunaan ulang (reuse) komponen perangkat lunak yang telah ada.
3. Pembangkitan kode program otomatis/semi otomatis.
4. Multiple team (banyak tim), tiap tim menyelesaikan satu tugas yang selevel tapi tidak sama. Banyaknya tim tergantung dari area dan kompleksitasnya sistem yang dibangun.

Jika keutuhan yang diinginkan pada tahap analisa kebutuhan telah lengkap dan jelas, maka waktu yang dibutuhkan untuk menyelesaikan secara lengkap perangkat lunak yang dibuat adalah berkisar 60 sampai 90 hari. Model RAD ini sebenarnya hampr sama dengan model waterfall,

bedanya siklus pengembangan yang ditempuh model ini sangat pendek dengan penerapan teknik yang cepat. Sistem dibagi-bagi menjadi beberapa modul dan dikerjakan beberapa tim dalam waktu yang hampir bersamaan dalam waktu yang sudah ditentukan. Model ini melibatkan banyak tim, dan setiap tim mengerjakan tugas yang selevel, namun berbeda. Sesuai dengan pembagian modul sistem.





Gambar 3.2 RAD Detail Model Proses

3.3. Instrumen Penelitian

Penelitian ini melibatkan beberapa instrumen yang mendukung terstrukturanya pengerjaan penelitian. Alat dan bahan yang di butuhkan dalam penelitian antara lain:

- Kamera
- Alat pengukur jarak dan skala
- Komputer yang memadai untuk pengerjaan penelitian

- Kamera

Kamera dibutuhkan untuk mengambil gambar gedung dari berbagai sudut pandang yang kemudian akan dimodelkan di software pembuat model 3D menjadi sebuah gedung virtual yang dibuat semirip mungkin dengan aslinya. Tak ada kamera khusus dalam pengambilan gambar, hanya saja gambar hasil *capture* harus memenuhi syarat yaitu gambar jelas dan menggambarkan gedung secara utuh dari tiap sisi gedungnya.

- Alat Pengukur Jarak

Alat ukur jarak mutlak harus ada dikarnakan perancangan gadung dan segala aspek yang terkait dalam lingkungan harus di ketahui jarak nya secara rinci. Perincian jarak tersebut dibutuhkan untuk perancangan fitur pencarian jalur terpendek dan perkiraan perancangan letak gedung secara presisi. Beruntung dalam penelitian ini saya mendapat *Master Plan* UPI, yaitu berupa draft peta UPI tampak atas yang menggambarkan UPI secara presisi dalam skala dari ukuran aslinya

- **Komputer**

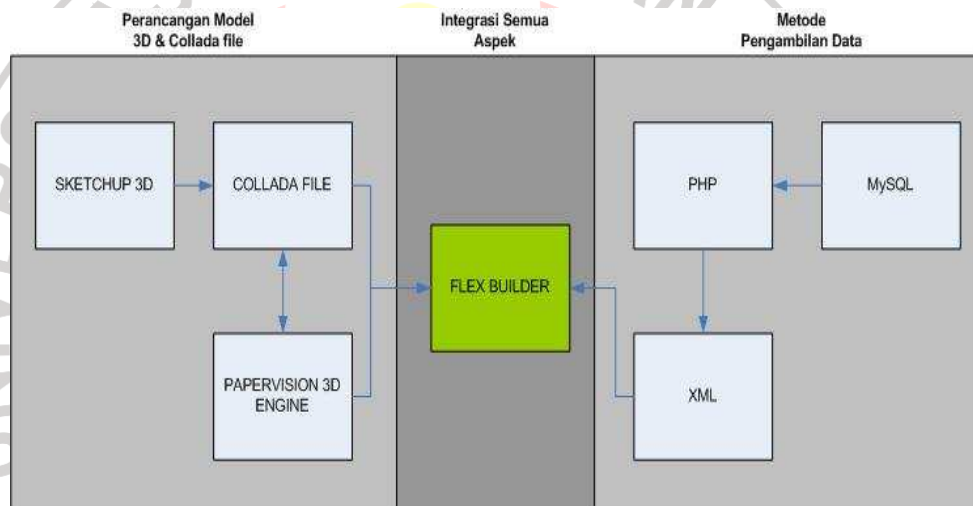
Komputer yang dibutuhkan harus memiliki spesifikasi minimal processor yang mendukung teknologi dual core dengan ram minimal 1 GB. Hal ini di karenakan perancangan objek 3D membutuhkan “tenaga” yang cukup menguras resource komputer itu sendiri. Demi kelancaran penelitian, komputer dengan spesifikasi yang disarankan adalah diatas spesifikasi yang telah disebutkan.



3.4. Implementasi

3.4.1 Prosedur Pengerjaan Penelitian

Penelitian Transformasi peta ini membutuhkan beberapa tahap pengerjaan mulai dari pekerjaan diluar komputer (terjuan langsung ke lokasi penelitian untuk mengambil data yang dibutuhkan) sampai merancangnya menjadi sebuah sistem yang terintegrasi dan utuh.

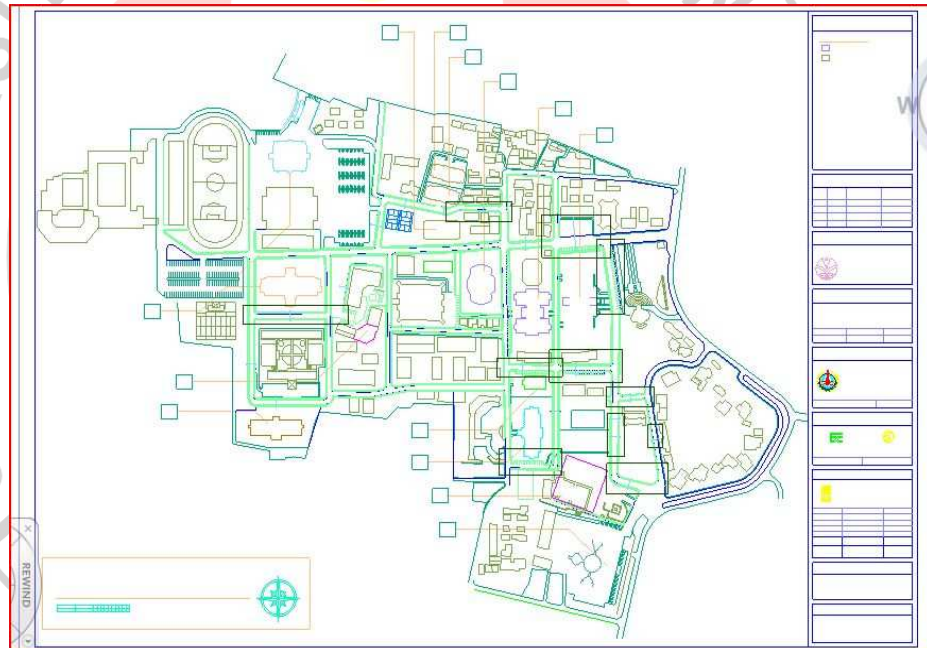


Gambar 3.3 Model Rancangan Program Keseluruhan

Adapun urutan pengerjaan yang harus dilalui adalah sebagai berikut:

1. Pengumpulan data yang berhubungan dengan data gedung, lokasi dan sekitarnya
2. Riset denah yang akan dibuat, perlu diperhatikan pula aspek skala semakin presisi semakin baik. Denah mengenai data lokasi yang telah terkumpul terlebih dahulu divisualisasikan menggunakan rancangan

gambar 2d, hal ini akan sangat memudahkan dalam perancangan selanjutnya (3D), karena kita bisa fokus terlebih dahulu kepada draf 2d yang lebih mudah dilakukannya koreksi apabila terjadi kesalahan, jadi tak akan membuang waktu pada saat pemodelan 3D. Untuk kasus pemodelan di UPI ini, saya menggunakan *Master Plan* (denah UPI secara keseluruhan dengan sekala presisi). Hal ini sangat membantu dalam peletakan posisi gedung serta jalan-jalan yang menghubungkannya.



Gambar 3.4 Master Plan UPI

3. Membuat rancangan virtual denah dalam modeler 3D dengan aplikasi yang telah dikuasai. Mungkin akan terlibat beberapa tool baik software

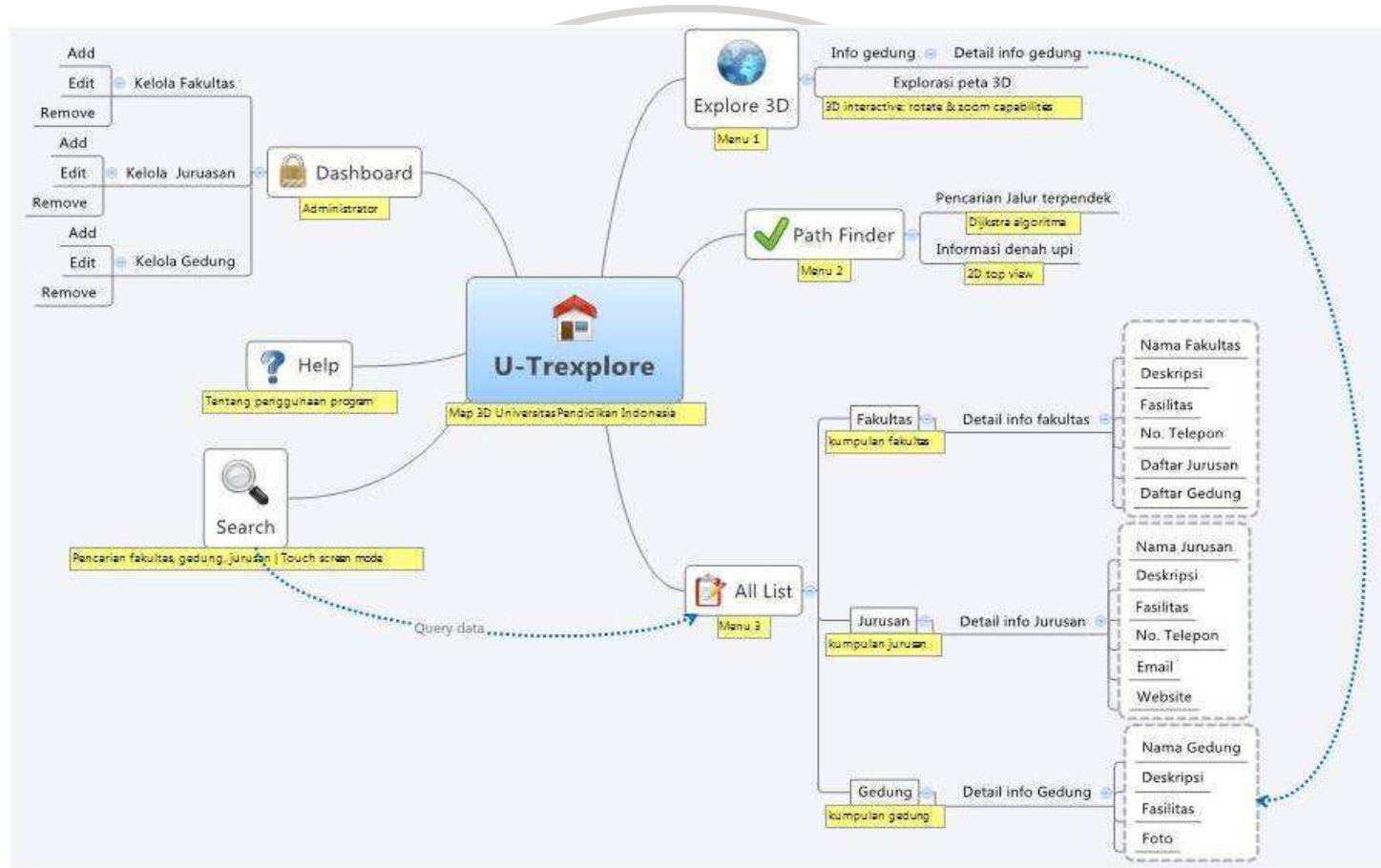
stand alone maupun plug-in yang bertujuan agar tingkat efektifitas perancangan menjadi tinggi dan flexible.

4. Pembangunan Interface Program di Flex yang disesuaikan dengan fitur-fitur yang akan ada dalam program ini, lima fitur besar itu adalah:
 - a. *explore 3D*,
 - b. *Find Path*
 - c. *All list*
 - d. *Search*
 - e. *Help*
5. Mengekpor hasil rancangan denah ke program *third party* yang menyediakan tools *programming* yang mendukung import file 3D. Dalam penelitian ini saya telah memutuskan dari sekian banyak aplikasi yang mendukung, saya memilih Flex sebagai tool main programming dan integrasi 3D. Hal tersebut dikarenakan telah saya temukannya plug-in (PaperVision3D) berupa class-class yang dapat mengintegrasikan model 3D kedalam flex yang nantinya akan di *coding* agar menjadi model 3D yang dapat di eksplor secara bebas.
6. Sinkronisasi pergerakan model 3D dengan status programing yang diberikan oleh program utama (Tempat dimana semua aspek diintegrasikan)
7. Perancangan dan Pembangunan Database yang disesuaikan dengan kebutuhan fitur. Peneliti menggunakan tool AMFPHP sebagai 'wali' untuk kebutuhan pengambilan data dari MySql menggunakan Flex.

8. Pemberian informasi pada tiap tempat yang masuk dalam area peta
9. Pembuatan Fitur *Find Path* atau pencarian jalur terpendek.
Menggunakan Algoritma Dijkstra
10. Tes semua aspek integritas, interface di Flex, programming pada model 3D, Database, Programming pada fitur *Find Path* dan fitur *Search*.
11. Compiling



Dibawah ini adalah bagan struktur keseluruhan dari program peta 3D upi:

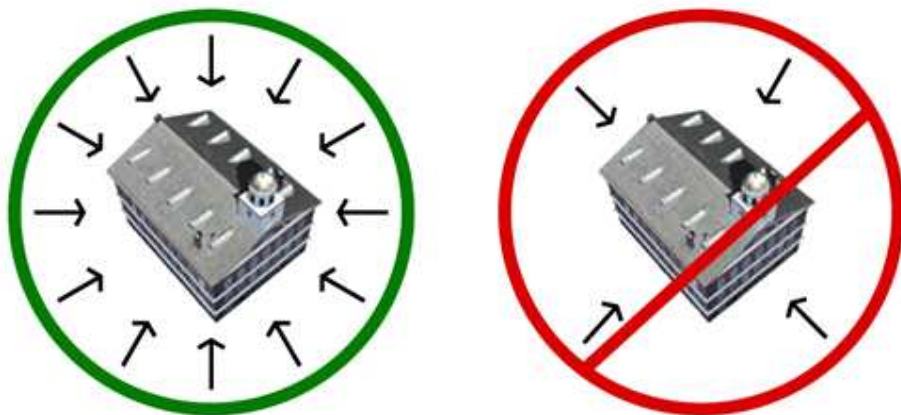


Gambar 3.5 Struktur Program

3.4.2 Pengumpulan Data

Pengumpulan data yang dimaksud meliputi:

- a. Data informasi mengenai gedung berupa informasi nama gedung, fakultas, fasilitas, jurusan, deskripsi, dan foto gedung.
- b. Data fisik gedung berupa *photo references*. Ini adalah tahapan pertama dalam pemodelan 3D. Foto yang gedung ini diperlukan dalam tahapan modeling objek.



Gambar 3.6 Pengambilan foto gedung dari berbagai sudut

Ada beberapa aspek yang perlu diperhatikan dalam pengambilan foto, diantaranya:

1. Ambil banyak gambar gedung dari berbagai sisi. Ini akan berguna untuk referensi modeling secara presisi dan keperluan *materialing*
2. Ambil juga gambar disekitar bangunan seperti pohon, mobil, dan lain-lain karena akan berguna dalam menciptakan *atmosphere* yang benar-benar merepresentasikan sebuah lokasi.

3. Ambil gambar 'detil/ part khusus bangunan' yang menjadi khas dari si bangunan.

Foto gedung yang peneliti ambil:



Gambar 3.7 Referensi pemodelan 3D dari foto . Tampak kanan



Gambar 3.8 Referensi pemodelan 3D dari foto . Tampak depan



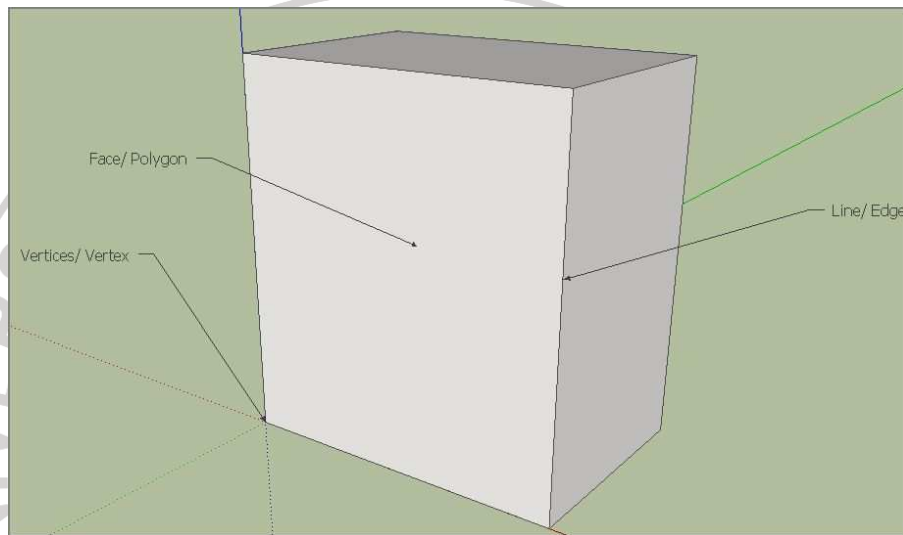
Gambar 3.9 Referensi pemodelan 3D dari foto . Tampak kiri



Gambar 3.10 Referensi pemodelan 3D

3.4.3 Modeling Objek 3D

Membuat model 3D yang benar berarti membuat model 3D yang dapat merepresentasikan bangunan dan juga memperhatikan efisiensi size dalam pemodelan. Model dengan geometri yang simple akan mudah dan cepat untuk di load program utama.



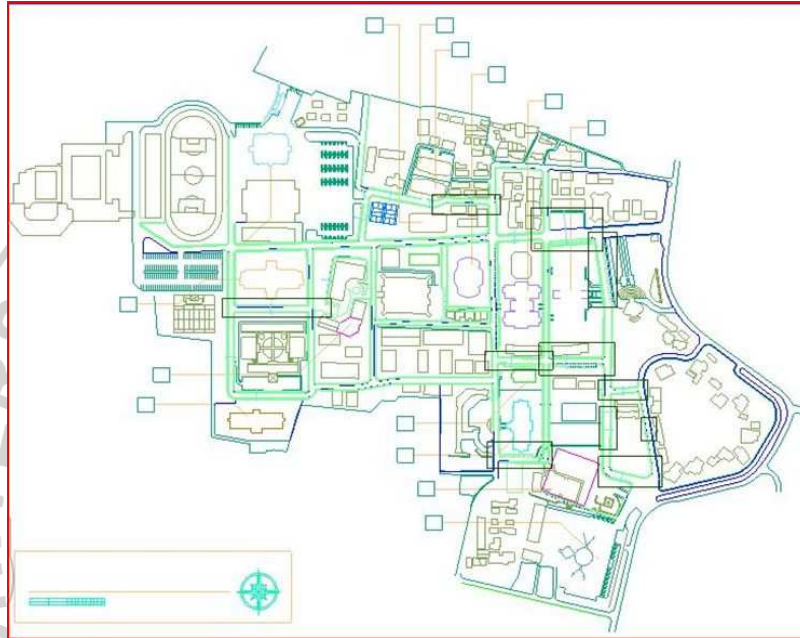
Gambar 3.11 Struktur objek 3D

Pengertian aspek pembangun suatu objek:

1. Face/ Polygon adalah sebuah plane yang terbentuk atas hasil keterhubungan *line*. Jumlah minimal line yg dibutuhkan untuk membentuk *face* adalah 3 line.
2. Vertices/ Vertex adalah sebuah titik ujung hasil pertemuan *line-line*
3. Line/Egde adalah sebuah garis yang tercipta karena keterhubungan 2 titik (Vertrice/ Vertex)

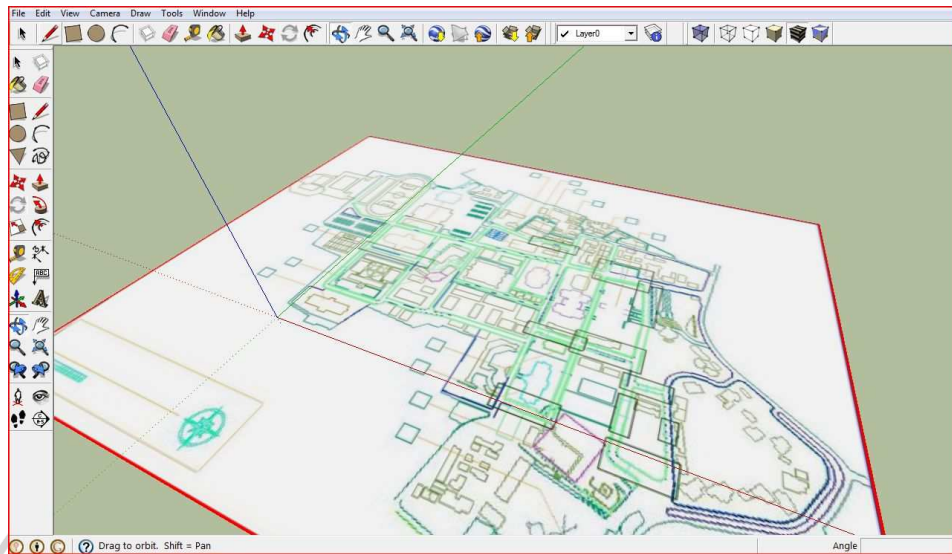
Untuk membuat sebuah model 3D yang ukurannya presisi, kita membutuhkan sebuah draf rancang bangun sebuah lokasi yang

menggunakan ukuran dengan skala berdasarkan ukuran sesungguhnya. Disarankan untuk menggunakan *Master Plan* atau *Blue Print* (Cetak biru) dari lokasi, ini akan sangat menghemat waktu penelitian mengenai daerah yang akan di ubah menjadi objek lokasi berbentuk 3D.



Gambar 3.12 Master Plan UPI

Hal pertama yang harus dilakukan dalam pemodelan presisi, adalah dengan menggunakan image reference sebagai acuan model dasar, yaitu menempelkan gambar Master Plan pada sebuah plane yang nantinya objek-objek gedung di bangun di atasnya sesuai dengan garis pada Master Plan tersebut.



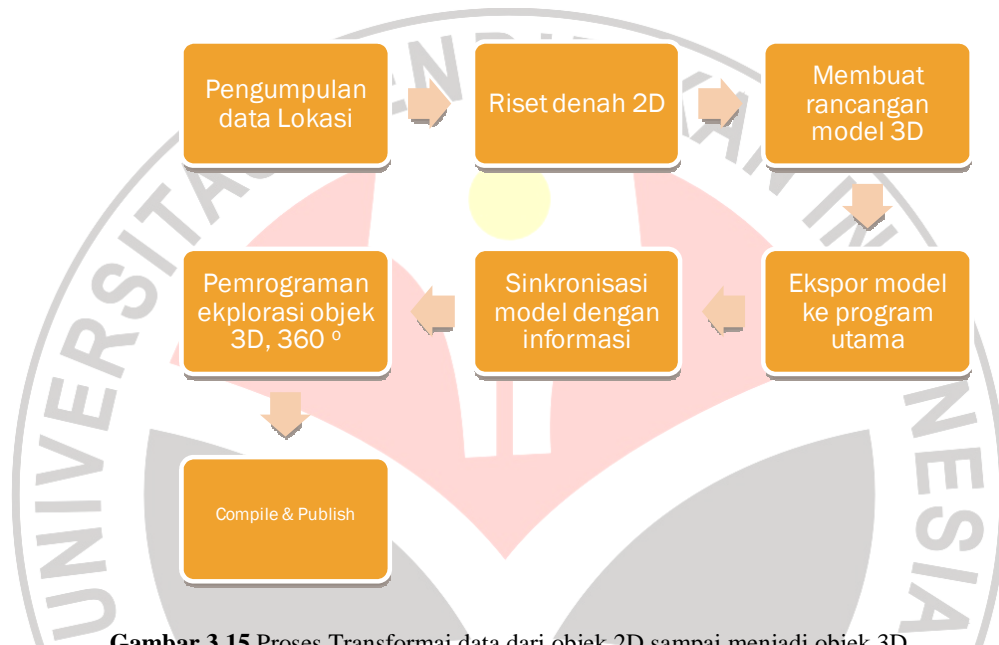
Gambar 3.13 Master plan UPI yang dijadikan dasar modeling 3D

Objek dibangun dari line sehingga memudahkan dalam pembentukan objek yang memiliki bentuk melengkung.



Gambar 3.14 Objek 3D (Plane) di bentuk oleh line

Tahapan selanjutnya yang akan dilakukan terhadap objek hasil modeling adalah sinkronisasi dengan program utama yaitu Flex Builder yang nantinya dilakukan tahap *programming* agar objek 3D statis dapat menjadi interaktif dan *explorable*. Tahapan-tahapannya sebagai digambarkan pada bagan dibawah ini:

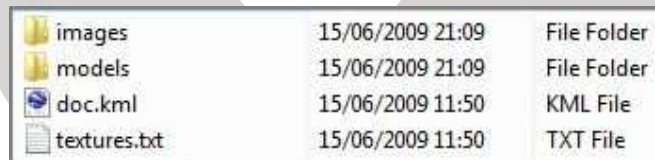


Gambar 3.15 Proses Transformasi data dari objek 2D sampai menjadi objek 3D

Objek 3D harus berjenis collada (*.dae - **digital asset exchange**) yang dapat di browse secara interaktif dengan engine flash player dan class Papervision.

Collada (**COLLAB**orative **DESIGN** Activity) adalah skema open standard XML untuk membuat sebuah objek 3D yang dapat di explore secara interaktif. Menurut pengembangnya [Khronos Group](#), Collada dapat digunakan secara gratis. Sketchup dapat mengkonvert hasil keluaran menjadi Collada file (*.dae). Ada 2 cara untuk mendapatkan objek 3D jenis ini, diantaranya:

1. Dengan mengexport langsung sebagai file collada. Program akan secara otomatis memasukan file DAE dan material yang digunakan pada objek. Namun fitur ini hanya bisa didapatkan pada program Sketchup berbayar.
2. Cara yang kedua adalah mengexportnya sebagai file Google Map (*.kmz), karena fitur *export* sebagai KMZ bisa dilakukan pada program sketchup yang tidak berbayar (gratis). Untuk mendapatkan file DAE dan Materialnya, kita cukup merubah ekstensi *.kmz menjadi *.zip, setelah itu ekstrak file, maka anda akan menemukan dua folder, satu file textures.txt dan file doc.kml, yang kita butuhkan hanya dua folder tersebut. Folder “model” yang bersisi file DAE dan satu folder berisi file “images” yang berisi material/ tekstur dari objek DAE tersebut.



images	15/06/2009 21:09	File Folder
models	15/06/2009 21:09	File Folder
doc.kml	15/06/2009 11:50	KML File
textures.txt	15/06/2009 11:50	TXT File

Gambar 3.16 Hasil ekstrak KMZ file menjadi ZIP file

Hasil keluaran tersebut di copy kedalam sebuah folder baru dalam Flex Project pembangunan peta 3D. Nama folder material harus sama dengan nama objek DAE nya. Jika nama objek DEA nya “jica.dae” maka folder material harus bernama “jica”. Nama material pun harus tepat seperti yang tertulis pada DAE file. Untuk meyakinkannya DAE file dapat dibuka dengan bantuan software editor teks dan temukan bagian:

```

<library_materials>
  <material id="material_0_1_8ID" name="material_0_1_8">
    <instance_effect url="#material_0_1_8-effect"/>
  </material>
  <material id="material_2_2_8ID" name="material_2_2_8">
    <instance_effect url="#material_2_2_8-effect"/>
  </material>
</library_materials>

```

Tag pada bagian “library_material” meminta referensi material berupa *image* kepada “library_images”. Nama texture0.jpg dan texture1.jpg adalah nama image yang dijadikan oleh objek sebagai material.

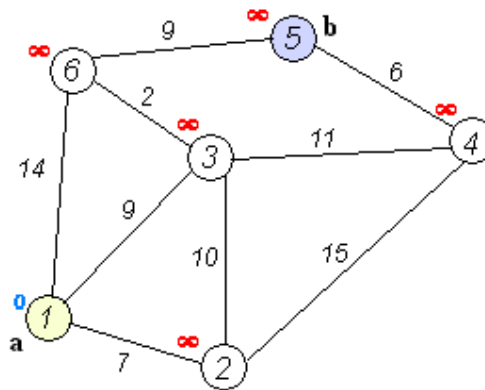
```

<library_images>
  <image id="material_0_1_8-image" name="material_0_1_8-image">
    <init_from>jicaTutupFace/texture0.jpg</init_from>
  </image>
  <image id="material_2_2_8-image" name="material_2_2_8-image">
    <init_from>jicaTutupFace/texture1.jpg</init_from>
  </image>
</library_images>

```

3.4.4 Metode Pencarian Jalur Terpendek (Dijkstra Algorithm)

Algoritma ini bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya. Misalkan titik menggambarkan gedung dan garis menggambarkan jalan, maka algoritma Dijkstra melakukan kalkulasi terhadap semua kemungkinan bobot terkecil dari setiap titik.



Gambar 3.17 Contoh keterhubungan antar titik dalam algoritma Dijkstra

Pertama-tama tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Inilah urutan logika dari algoritma Dijkstra:

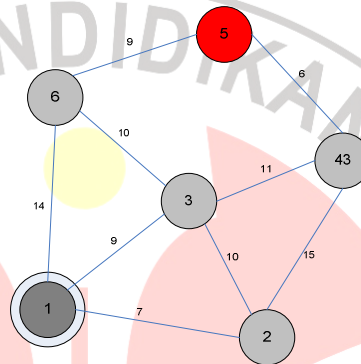
1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi)
2. Set semua node “Belum terjamah” dan set node awal sebagai “Node keberangkatan”
3. Dari node keberangkatan, pertimbangkan node tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh,

jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke node C berjarak 2, maka jarak ke C melewati B menjadi $6+2=8$. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

4. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai “Node terjamah”. Node terjamah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set “Node belum terjamah” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan lanjutkan dengan kembali ke step 3

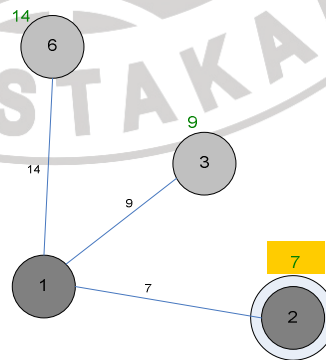
Dibawah ini penjelasan langkah per langkah pencarian jalur terpendek secara rinci dimulai dari node awal sampai node tujuan dengan nilai jarak terkecil.

1. Node awal 1, Node tujuan 5. Setiap edge yang terhubung antar node telah diberi nilai



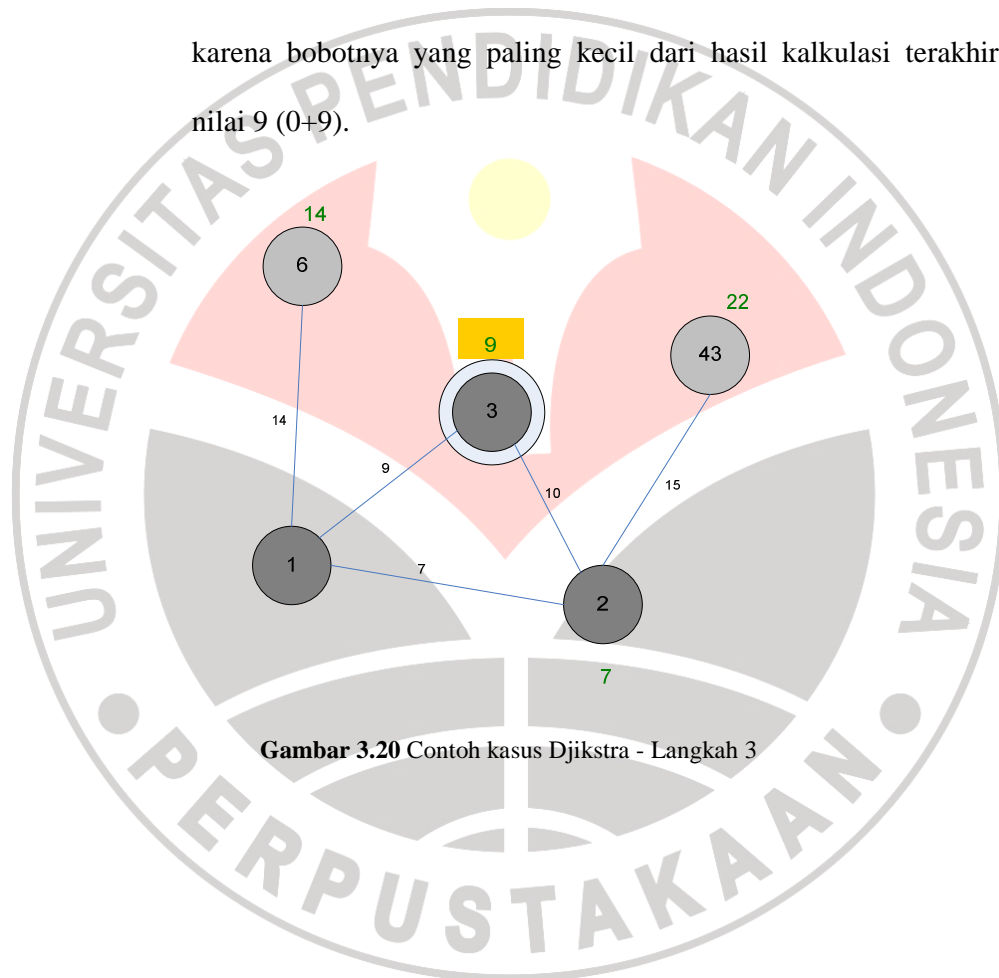
Gambar 3.18 Contoh kasus Dijkstra - Langkah 1

2. Dijkstra melakukan kalkulasi terhadap node tetangga yang terhubung langsung dengan node keberangkatan (node 1), dan hasil yang didapat adalah node 2 karena bobot nilai node 2 paling kecil dibandingkan nilai pada node lain, nilai = 7 ($0+7$).



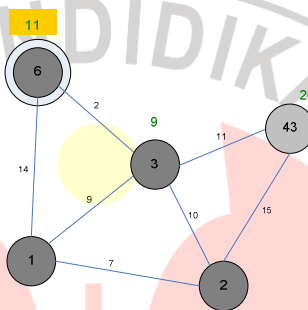
Gambar 3.19 Contoh kasus Dijkstra - Langkah 2

3. Node 2 diset menjadi node keberangkatan dan ditandai sebagai node yang telah terjamah. Dijkstra melakukan kalkulasi kembali terhadap node-node tetangga yang terhubung langsung dengan node yang telah terjamah. Dan kalkulasi dijkstra menunjukkan bahwa node 3 yang menjadi node keberangkatan selanjutnya karena bobotnya yang paling kecil dari hasil kalkulasi terakhir, nilai 9 ($0+9$).



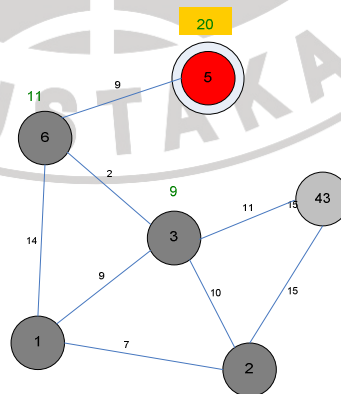
Gambar 3.20 Contoh kasus Dijkstra - Langkah 3

4. Perhitungan berlanjut dengan node 3 ditandai menjadi node yang telah terjamah. Dari semua node tetangga belum terjamah yang terhubung langsung dengan node terjamah, node selanjutnya yang ditandai menjadi node terjamah adalah node 6 karena nilai bobot yang terkecil, nilai 11 ($9+2$).



Gambar 3.21 Contoh kasus Dijkstra - Langkah 4

5. Node 6 menjadi node terjamah, dijkstra melakukan kalkulasi kembali, dan menemukan bahwa node 5 (node tujuan) telah tercapai lewat node 6. Jalur terpendeknya adalah 1-3-6-5, dan nilai bobot yang didapat adalah 20 ($11+9$). Bila node tujuan telah tercapai maka kalkulasi dijkstra dinyatakan selesai.



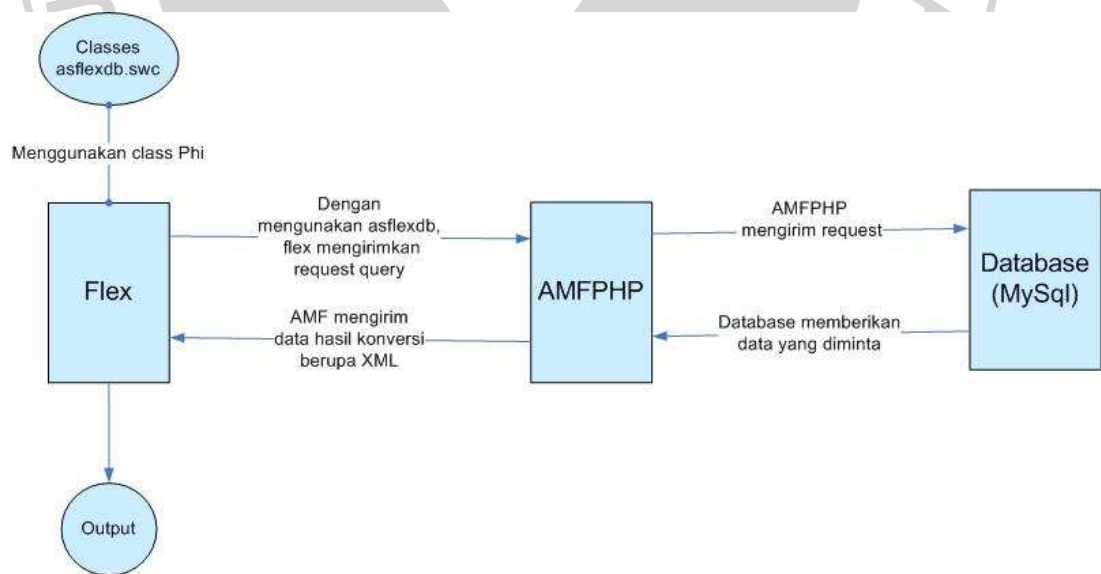
Gambar 3.22 Contoh kasus Dijkstra - Langkah 5

3.4.5 Metode Pengambilan data MySQL Database dengan Flex Builder

Flex tidak dapat secara langsung mengambil data dari database, dibutuhkan sebuah “jembatan” yang menjembatani lalu lintas data antara flex dengan database. Dari beberapa tools yang dapat menjembatani flex dengan MySQL, peneliti memilih AMFPHP dikarenakan kemudahan dan fitur *Browser* dari AMFPHP yang dapat memantau lintas data.

Untuk lebih memudahkan peneliti menggunakan *Class Library* “asflexdb.swc” yang memungkinkan pengerjaan menjadi lebih cepat dibandingkan harus membuat method-method PHP untuk amfphp class secara manual satu per satu. Dengan classes asflexdb Flex dapat melakukan query langsung terhadap database.

Dibawah ini adalah bagan dari keterhubungan Flex-MySQL:



Gambar 3.23 Skema pengambilan data dari Database

Pertama yang harus dilakukan adalah meng *include* classes asflexdb ke folder libs pada flex project. Class ini berfungsi sebagai penghubung Flex dengan MySql yang melewati AMFPHP sebagai control dari lalu lintas data. Asflexdb ini berisi kumpulan class bernama “phi” yang berisi:

1. Class Controls,
2. Class Db,
3. Class Interfaces.

Untuk melakukan pengambilan data dari MySql, kita cukup menggunakan dua class yaitu class “interfaces” dan class “db”. Class “db” berisi method-method yang memungkinkan flex melakukan koneksi terhadap MySql, sedangkan class “interface” memungkinkan flex untuk mengolah data query dan menampilkannya melalui bantuan *component* pada Flex.