# PEMBUATAN SISTEM VISUALISASI PETA PADA PONSEL DENGAN MEMANFAATKAN TEKNOLOGI XSLT (*EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMATION*)

## TUGAS AKHIR





Disusun Oleh :

RADITYA ARIF PERDANA
5202 100 017

**PROGRAM STUDI SISTEM INFORMASI**
**FAKULTAS TEKNOLOGI INFORMASI**
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**
**SURABAYA**
**2006**

# PEMBUATAN SISTEM VISUALISASI PETA PADA PONSEL DENGAN MEMANFAATKAN TEKNOLOGI XSLT (*EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMATION*)

# TUGAS AKHIR

**Diajukan untuk Memenuhi Sebagian Persyaratan**

**Memperoleh Gelar Sarjana Komputer**

**Pada**

**Program Studi Sistem Informasi**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya**

**Mengetahui/Menyetujui**

**Dosen Pembimbing**


**Febriliyan Samopa, S.Kom, M.Kom**

**NIP. 132 206 858**

**SURABAYA**

**AGUSTUS 2006**

# ABSTRAK

Pesatnya perkembangan teknologi *mobile programming* saat ini telah mendorong banyaknya aplikasi baru yang muncul pada ponsel. Visualisasi peta salah satunya, mengharuskan pengguna untuk memiliki peralatan tambahan berupa GPS yang mahal. Untuk itu penelitian ini akan berusaha mengembangkan aplikasi serupa pada ponsel, yang meskipun fitur-fiturnya nanti tidak akan selengkap jika dengan menggunakan tambahan GPS, tetapi diharapkan dapat memberikan manfaat bagi pengguna ponsel secara umum tanpa harus menambah peralatan apapun lagi.

Pengerjaan penelitian ini diawali dengan pengubahan format data spasial pada *server* yang akan memermudah pemrosesan data dari *client*. Data spasial dapat diekstrak dan diubah menjadi format XML dengan menggunakan *library* dari Deegree dan Saxon. Setelah diproses, data dengan format XML dapat dikonversi menjadi gambar vektor SVG dengan menggunakan XSLT. XSLT adalah dokumen berbasis XSL yang dapat berfungsi untuk mentransformasikan data XML menjadi elemen-elemen grafis SVG. Terakhir gambar vektor SVG diubah ke gambar *bitmap* dengan *library* dari Batik.

Penelitian ini menghasilkan aplikasi yang mampu menghadirkan pada penggunanya, visualisasi peta sebuah daerah beserta fitur yang telah dipilihnya, pada ponselnya. Gambar tersebut dihasilkan dari format data spasial yang dikonversi ke format data XML dan kemudian ditransformasikan ke format *bitmap*. Proses konversi ini berhasil dilakukan tanpa mengubah informasi yang terkandung pada format data spasial.

**Kata kunci**: *visualisasi peta, data spasial, XML, SVG, XSLT, bitmap, Deegree, Saxon, Batik*

## *ABSTRACT*

*The enormous growth of mobile programming tecnology nowdays has been driven a lot of new application to be built on cellphones. One of them, map visualization, oblige the user to have an extra device, like a GPS that is very expensive. For that reason, this research is done to develop similar application on cellphones that although, not going to have features as complete as like it have been with a GPS, but expected to provide use for cellphone users generally without having to add anymore equipment.*

*This research begin by converting spatial data format on the server to make data from the client easier to process. The spatial data could be extracted and changed into XML with the use of library from Deegree and Saxon. After being processed, the XML formatted data is converted into SVG vector image, utilizing XSLT. XSLT is an XSL based document that can be used to transformed XML data into SVG graphical elements. Last, the SVG vector image is changed into bitmap image using library from Batik.*

*The end result of this research is an application that enables the user to see visualization of the map consisting region and feature that has been chosen before, on his/her cellphone. The image is produced from the spatial data format converted into XML and then transformed into bitmap format. This converting process is successfully done without altering any information in the spatial data*



**Keywords:** *map visualization, spatial data, XML, SVG, XSLT, bitmap, Deegree, Saxon, Batik*

# KATA PENGANTAR

ALHAMDULILLAH...

Selesai sudah akhirnya salah satu sub bab dalam Bab Kehidupan Kuliah ini, Tugas Akhir, sebagai salah satu syarat mendapat tambahan "embel-embel" di belakang nama yang telah ada, dengan judul seperti yang dapat dibaca di bawah:

**"Pembuatan Sistem Visualisasi Peta Pada Ponsel Dengan Memanfaatkan Teknologi XSLT (*Extensible Stylesheet Language Transformation*)"**

Tentu semua ini tidak akan dapat terwujud tanpa ridha-Nya, semua ini tidak akan dapat terwujud tanpa campur tangan dan perhatian yang telah diberikan semua yang telah hadir dalam hidup penulis. Untuk itu semua, terima kasih penulis sampaikan kepada:

1. Kedua-orangtua penulis, atas segala yang tidak akan pernah dapat terganti, kecuali dari-Nya.
2. Dek Angga, jangan ngerepotin Mama-Papa terus ya...
3. Bapak Febriliyan Samopa, M.Kom, selaku dosen pembimbing yang telah banyak memberikan pikiran dan waktunya yang berharga.
4. Mas Benny dan Milan, untuk TA dan NIMA-nya.
5. Kidd, untuk segala bantuan teknisnya.
6. Bapak Aris Tjahyanto, M.Kom,, selaku dosen wali penulis yang memberikan masukan dan nasehat.
7. Dosen-dosen Sistem Informasi: Ibu Erma Suryani, MT, Bpk. Arif Djunaidy, PhD., Wiwik Anggraeni, M.Kom, Bpk. Rully Soelaiman, M.Kom, Bpk Ir. Khakim Ghozali, Bpk. Mudjahidin, MT, Bpk. Faizal Johan A, S.Kom, Bpk. Edwin Riksakomara, MT, Bpk. Ahmad Holil N.A., M.Kom, Bpk Bambang Setiawan, MT, Bpk. Rully Agus H., S.Kom.
8. Dosen-dosen ITS yang telah mendidik penulis selama perkuliahan.
9. Mas Bambang sebagai administrator lab SI, terima kasih atas kemudahan-kemudahan yang telah banyak diberikan.

10. Karyawan Sistem Informasi dan FTIf pada umumnya: Mas Kadir, Mbak Anita Pak Yudi, Pak Narno, Bu Tutik, Pak Muin, Pak Karmono, Pak Soleh, Mas Sugeng, Mbak Eva, dkk.
11. Swing, Trix #7, Sug4, Dion, Pepenk, dan Cahyo, yang telah rela memberikan sepetak kamarnya untuk melindungi penulis dari dingin, terik dan badai.
12. "Saudara" 02 yang lain, Goz, Sid, Hand-Dick, Erick, Cak Son, Willy, Pras, Ardee, Ayu', Unee, Prima, dll (Sory yo Cuma tak tulis "dll"... ), atas segala kenangan ini (kapan S.E. lagi?).
13. Mas-mbak 01 dan Adek-adek 03, 04, 05, yang telah melengkapi segala pengalaman 4 tahun terakhir ini (Qon, Ci, Ma, tak usahain ga jaim lagi).
14. Nu & Sus, My Dearest Friends, thanks for all the movies, songs, juices, and nights we spent with all the mosquitoes...
15. Last but not least, bagi dia yang telah membawa pijar dalam hidupku..., semoga selalu diberi yang terbaik dari-Nya.

Terima kasih semuanya...

Akhirnya, sebagaimana manusia yang tidak sempurna, penulis sadar bahwa karya ini tidak lepas dari kecacatan di sana-sini. Tetapi besar harapan penulis, agar ilmu yang terkandung dapat bermanfaat kepada pembaca.

Sidoarjo, Agustus 2006

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I

# PENDAHULUAN

# BAB I
# PENDAHULUAN

## 1.1 LATAR BELAKANG

Seiring dengan semakin pesatnya perkembangan teknologi saat ini dan tuntutan mobilitas yang semakin tinggi, kebutuhan akan ponsel telah menjadi salah satu kebutuhan dasar, yaitu barang yang umum dimiliki oleh semua kalangan, dan tidak lagi dianggap sebagai barang yang mewah. Cukup dengan memilliki ponsel? Ternyata belum! Kemobilitasan yang telah menjadi aspek penting dalam tiap sisi kehidupan manusia modern sekarang ini juga menuntut pihak pengembang untuk semakin proaktif dalam melahirkan inovasi-inovasi baru di tiap produknya jika ingin bertahan.

Inovasi-inovasi yang terus bermunculan ini pada akhirnya akan menghasilkan berbagai macam fitur baru di ponsel. Setiap fitur memiliki manfaat yang berbeda-beda sesuai dengan kebutuhan manusia akan hal-hal yang praktis. Pemesanan tiket, melihat jadwal bioskop, pembayaran tagihan, dan visualisasi peta adalah beberapa contoh dari banyaknya fitur dan aplikasi baru yang dibuat dengan berbasiskan teknologi *mobile*.

Khusus untuk fitur yang disebutkan terakhir (visualisasi peta), pengguna umumnya diharuskan memiliki peralatan tambahan berupa GPS (*Global Positioning System*). Untuk mendapatkan peralatan tambahan ini kita diharuskan untuk merogoh kantong kita cukup dalam (berkisar antara $170 - $1,100).

Berdasarkan latar belakang di atas penulis akan mengembangkan sistem visualisasi peta pada ponsel yang walaupun fitur-fitur yang ditawarkan tidak akan selengkap dengan apa yang dapat diberikan aplikasi dengan tambahan GPS, tetapi diharapkan dapat memberikan manfaat bagi pengguna ponsel secara umum tanpa harus menambah peralatan apapun lagi.

## 1.2 TUJUAN PEMBUATAN TUGAS AKHIR

Tujuan dari tugas akhir ini adalah pembuatan sistem visualisasi peta pada ponsel. Diharapkan nantinya aplikasi ini dapat memudahkan pengguna ponsel untuk melihat peta sebuah daerah beserta fitur-fiturnya (letak rumah sakit, stasiun, *zoom in*, dll.) tanpa tambahan alat (mis: GPS).

## 1.3 MANFAAT

Manfaat yang diberikan oleh tugas akhir ini adalah sebagai berikut:

- Memberikan kemudahan bagi pengguna ponsel untuk melihat peta sebuah daerah beserta fitur-fiturnya tanpa tambahan alat.
- Memberikan kontribusi dalam pengembangan teknologi *mobile* dan GIS *programming*.

## 1.4 PERMASALAHAN

Permasalahan yang diangkat dalam tugas akhir ini adalah sebagai berikut:

- Bagaimana mengonversi format data spasial menjadi XML.
- Bagaimana mengonversi format XML menjadi *bitmap*.
- Bagaimana memroses informasi yang diberikan *client* (ponsel) untuk diaplikasikan pada format data spasial di *server*.

## 1.5 BATASAN MASALAH

Dari permasalahan yang telah disebutkan di atas, maka batasan-batasan dalam tugas akhir ini adalah sebagai berikut:

- Format data spasial yang digunakan pada *server* adalah *shapefile* dari ESRI.
- Format gambar yang digunakan pada *client* (ponsel) adalah .png.
- Studi kasus akan menggunakan peta Kota Madiun.
- Aplikasi pada *server* akan dikembangkan dalam *platform* Java dengan menggunakan JDK 1.5.0 dan tambahan *library* dari Deegree, Saxon, dan Batik.

- Aplikasi pada *client* (ponsel) akan dikembangkan dalam *platform* Java dengan menggunakan J2ME Wireless Toolkit 2.2 dan *library* MIDP 2.0.
- Ponsel yang digunakan adalah Motorola C651.

## 1.6 METODOLOGI PENYUSUNAN TUGAS AKHIR

Pembuatan tugas akhir ini akan dibagi menjadi beberapa tahap pengerjaan sebagai berikut:

### 1.6.1 Studi Literatur

Tahap ini dilakukan untuk mendapatkan pemahaman mengenai bidang-bidang sebagai berikut yang akan mendukung dalam pembuatan tugas akhir ini nantinya:

- teknik pemrograman dalam J2ME, terutama yang berkaitan dengan komunikasi jaringan dan pengunduhan gambar,
- konsep-konsep dasar yang perlu diketahui dalam GIS, termasuk di dalamnya format *shapefiles* dari ESRI,
- pencarian dan pemahaman penggunaan *library-library* tambahan yang diperlukan,
- serta teknologi XSLT yang akan digunakan dalam pengonversian format.

### 1.6.2 Perancangan Aplikasi

Hal yang dilakukan pada tahap ini adalah merancang model aplikasi yang akan dibuat. Termasuk dalam tahapan ini adalah gambaran dasar metode pengonversian format pada sisi server dan komunikasi jaringan antara *server-client*, serta desain antar-muka aplikasi.

### 1.6.3 Pembuatan Aplikasi

Tahap ini merupakan tahap pembuatan dan pengembangan aplikasi berdasarkan pada rancangan dan gambaran dasar yang telah dibuat di tahapan sebelumnya. Sistem visualisasi peta ini nantinya akan dikembangkan dengan

menggunakan simulator ponsel J2ME Wireless Toolkit 2.2 keluaran Sun Microsystems, Inc.

### 1.6.4 Uji Coba dan Evaluasi

Pada tahap ini semua fungsi yang terdapat pada aplikasi akan diujicobakan langsung menggunakan ponsel.

### 1.6.5 Penyusunan Buku Tugas Akhir

Tahap terakhir ini merupakan dokumentasi pelaksanaan tugas akhir. Diharapkan, buku tugas akhir ini akan bermanfaat bagi pembaca yang ingin mengembangkan aplikasi ini lebih lanjut atau sekedar memelajari pengetahuan yang terdapat di dalamnya.

## 1.7 SISTEMATIKA PENULISAN

Sistematika penulisan Buku Tugas Akhir ini akan dibagi menjadi enam bab yang terurai sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini akan dijelaskan tentang latar belakang, tujuan, manfaat, permasalahan, batasan masalah, serta metodologi penyusunan buku tugas akhir.

BAB II DASAR TEORI

Pada bab ini akan dibahas teori-teori yang menjadi dasar dari pembuatan tugas akhir yang akan meliputi pengertian tentang SIG, *Shapefile*, J2ME, XML, XML *Schema*, XML *Link*, XML *Pointer*, XSL, XSLT, GML, SVG, dan *library-library* dari *Java Open Source* yang digunakan.

BAB III PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas perancangan dari sistem visualisasi peta yang meliputi aspek arsitektur *client-server*, data, proses, dan antar-muka perangkat lunak. Bab ini dapat memberikan gambaran dasar mengenai perangkat lunak yang akan dikerjakan.

BAB IV IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini akan diberikan realisasi dari rancangan keseluruhan proses yang terdapat pada aplikasi, baik di sisi *server* maupun di sisi *client*, serta bentuk implementasi antar-muka aplikasi di sisi *client*.

BAB V UJI COBA DAN EVALUASI

Pada bagian ini akan dijelaskan tentang pelaksanaan uji coba terhadap proses yang dikerjakan perangkat lunak beserta evaluasinya. Uji coba ini dilakukan berdasarkan permasalahan yang diangkat dalam tugas akhir dan jawabannya.

BAB VI PENUTUP

Pada bab ini akan diberikan simpulan berupa jawaban dari permasalahan yang menjadi fokus utama pengerjaan tugas akhir beserta saran yang dapat dijadikan acuan pada pengembangan selanjutnya.

# BAB II

# DASAR TEORI

# BAB II
# DASAR TEORI

Pada bab ini akan dibahas teori-teori yang menjadi dasar dari pembuatan tugas akhir. Pembahasan akan meliputi pengertian tentang SIG, *Shapefile*, J2ME, XML, XML *Schema*, XML *Link*, XML *Pointer*, XSL, XSLT, GML, SVG, dan *library-library* dari *Java Open Source* yang digunakan.

## 2.1 SIG (SISTEM INFORMASI GEOGRAFIS)

SIG (Sistem Informasi Geografis) merupakan sebuah sistem untuk mengatur, menganalisis, dan menampilkan informasi geografis yang diwakili oleh seperangkat informasi. [6]. Seperangkat informasi tersebut antara lain:

- Peta dan *Globe*, penampilan data geografis secara interaktif yang digunakan untuk menjawab pertanyaan, menampilkan hasil, dan digunakan sebagai perwakilan terhadap dunia nyata.
- Set data geografis, berisi tentang penyimpanan data geografis baik dalam bentuk *database* maupun berbasis dokumen yang berupa fungsi, jaringan, daerah, jajak pendapat dan atribut.
- Pemrosesan dan model diagram alur, seperangkat fungsi *geoprosessing* yang digunakan untuk otomasi dan pengulangan pekerjaan serta untuk analisa.
- Model data, data set SIG lebih dari sekedar DBMS (*Database Management System*). Data set tersebut menggabungkan perilaku lebih dan lengkap seperti sistem informasi lainnya. Skema, perilaku, dan aturan lengkap dari set data geografis memiliki peranan penting dalam SIG.
- *Metadata*, merupakan catatan yang memberikan penjelasan untuk elemen yang lain. Sebuah katalog dokumen memungkinkan pengguna untuk mengatur, menemukan dan mencapai akses ke informasi geografis.

Pada awal masa penggunaan SIG, para pengembang berkonsentrasi pada kompilasi data dan fokus pada pengembangan aplikasi, menghabiskan waktu untuk membuat *database* SIG dan menyusun informasi geografis.

Secara berangsur-angsur, pengguna SIG mulai menggunakan dan mengeluarkan koleksi informasi dalam berbagai aplikasi dan pengaturan SIG. Pengguna memergunakan SIG *workstation* untuk mengompilasi set data geografis, membangun diagram alur untuk kompilasi data dan pengawasan kualitas, penyusunan peta dan pemodelan analitis, serta mendokumentasikan pekerjaan dan metode yang digunakan.

Hal ini menimbulkan pandangan tradisional pada pengguna SIG terhadap pengembang *workstation* untuk fungsi keilmuan yang terhubung pada set data dan *database*. *Workstation* memiliki aplikasi SIG secara lengkap dengan logika SIG terbaru dan peralatan untuk menyelesaikan hampir semua keperluan SIG.

Konsep perangkat lunak SIG ini terbukti kurang memiliki nilai kurang dan diadopsi secara luas oleh pengembang SIG dalam kurang lebih 200.000 organisasi di dunia. Pada kenyataannya, sistem komputasi *client-server* telah memiliki keberhasilan yang tinggi dibandingkan pemikiran SIG pada konteksnya saja. Bagaimanapun juga, visi dari SIG terus berkembang.

Pengembangan komputasi terkini, perkembangan dari internet, peningkatan dalam teknologi DBMS, pemrograman berbasis obyek, komputasi *mobile* dan adopsi SIG secara luas telah memimpin perubahan visi dan aturan dari SIG.

Selanjutnya pada aplikasi SIG, perangkat lunak SIG dapat diletakkan pada *server* aplikasi secara terpusat dan *web server* untuk menghadirkan kemampuan SIG pada semua pengguna dalam jaringan. Pengguna SIG dapat terhubung pada pusat *server* SIG menggunakan cara tradisional, aplikasi SIG lainnya, misalkan *web browser*, aplikasi khusus, peralatan komputasi *mobile*, dan peralatan digital lainnya. Visi platform SIG ini yang sekarang berkembang.

### 2.1.1 *Shapefile*

*Shapefile* adalah salah satu format data spasial untuk fitur data lengkap dengan atributnya dimana penghapusan fitur dan pengubahan nilai atribut dapat dilakukan. Format jenis ini menyimpan bentuk geometri dan informasi atribut mengenai fitur spasial dalam sebuah set data.

Karena *shapefile* tidak memerlukan pemrosesan struktur data topologis dan hanya menangani fitur-fitur yang tidak kontinyus maka format ini memiliki beberapa kelebihan bila dibandingkan dengan sumber data lain, seperti:

- kemampuan menggambar yang lebih cepat,
- kemampuan pengubahan nilai,
- kapasitas penyimpanan yang lebih kecil,
- dan kemudahan dalam proses *read-write*.

Dengan adanya kelebihan-kelebihan di atas maka *shapefiles* digunakan sebagai format data spasial dalam pengerjaan tugas akhir ini.

## 2.2 J2ME (JAVA 2 MICRO EDITION)

Java 2 Micro Edition (J2ME) adalah kumpulan teknologi dan spesifikasi yang memungkinkan developer untuk membangun dan mengembangkan aplikasi menggunakan teknologi Java yang sesuai dengan kebutuhan dan kemampuan dari *mobile device*. J2ME dibagi menjadi 3 bagian, yaitu konfigurasi, *profile*, dan *package* tambahan.

### 2.2.1 Konfigurasi (*Configuration*)

Konfigurasi adalah spesifikasi yang menjelaskan tentang *virtual machine* dan seperangkat *library* yang menyediakan API-API yang diperlukan pada kategori *device* tertentu. Saat ini, ada dua konfigurasi J2ME, yaitu *Connected Limited Device Configuration* (CLDC) dan *Connected Device Configuration* (CDC). CLDC adalah konfigurasi J2ME yang digunakan pada *device* nirkabel dengan kemampuan yang terbatas, seperti ponsel dan PDA. Sedangkan CDC adalah konfigurasi J2ME yang digunakan untuk *device* yang kemampuannya memori dan kekuatan prosesnya lebih besar, seperti *smartphone*.

### 2.2.2 Profil (*Profile*)

Sebuah profile adalah seperangkat API dengan tingkatan lebih tinggi yang mendefinisikan model siklus hidup aplikasi, antarmuka *user*, *persistent storage*, dan akses ke properti *device* tertentu. Profil melengkapi konfigurasi dengan menambahkan API yang lebih spesifik untuk kategori *device* yang spesifik.

### 2.2.3 *Package* Tambahan (*Optional Package*)

*Package* tambahan memerluas kemampuan J2ME dengan menambahkan fungsionalitas-fungsionalitas yang berhubungan dengan CLDC, CDC, maupun profil terkait. *Package* tambahan menyediakan API standar yang dibuat untuk memenuhi kebutuhan aplikasi yang sangat spesifik, seperti koneksi database, *wirelesss messeging*, multimedia, grafis 3D, dan *Web Service*.

## 2.3 XML (EXTENSIBLE MARKUP LANGUAGE)

Pada awal digunakannya dokumen elektronik, hal yang lebih ditekankan adalah bagaimana dokumen disajikan daripada struktur dokumen dan pengartian. Troff dan TeX, dua bahasa pengubah format pertama, berhasil melakukan perubahan format dokumen dengan baik, akan tetapi sangat kurang dalam struktur dokumen. Konsekuensinya adalah dokumen memiliki keterbatasan dalam penyajian di layar atau ketika dicetak serta kesulitan dalam penulisan program untuk melakukan pencarian, mengeluarkan informasi, melakukan *cross-references* secara elektronik atau penggunaan ulang dokumen untuk aplikasi yang berbeda. Pemrograman dengan tag yang sudah dideskripsikan terlebih dahulu, mampu menyelesaikan permasalahan ini.

Pada akhir 1960-an, proyek *GenCode* dibangun untuk melakukan pengubahan format dari dokumen yang berbeda dengan tag umum dan untuk menggabungkan dokumen dari berbagai macam penggalan. Selanjutnya yang dikembangkan adalah GML (*Generalized Markup Language*) oleh IBM, yang mana memberikan solusi terhadap pengodean dokumen untuk penggunaan dengan berbagai macam subsistem informasi. Dokumen yang dikodekan dalam bahasa

markup ini dapat diketik ulang, diubah formatnya, dan dilakukan pencarian menggunakan aplikasi yang berbeda.

Kemudian pada tahun 1985, mulai dibangun SGML (*Standard Generalized Markup Language*) yang menjadikan GML suatu standar dalam pengolahan dokumen elektronik. Pada dasarnya SGML dibangun sebagai alat bantu untuk mengembangkan bahasa markup yang lebih khusus. SGML memiliki sintaks yang kompleks dan menggunakan banyak parameter, sehingga menjadi kompleks dan mahal untuk pengolahan dalam skala kecil.

Pada awal 1990, muncul HTML (*Hypertext Markup Language*) yang didesain lebih kompak dan efisien dibandingkan SGML. Akan tetapi memiliki keterbatasan dalam pengolahan dokumen yang digunakan untuk berbagai macam keperluan. Tag-tag HTML tidak dapat didefinisikan kembali untuk keperluan khusus sehingga beberapa orang mencoba untuk menggunakan SGML dalam *web*. Akan tetapi SGML merupakan bahasa yang sangat kompleks bila dibandingkan dengan *web browser* yang memiliki kemampuan terbatas. Muncullah satu bahasa yang memiliki kemampuan SGML akan tetapi lebih kompak, yaitu XML (*Extensible Markup Language*). Pada pertengahan 1990 W3C (*World Wide Web Consortium*) mengeluarkan satu standar dalam XML yang mengkombinasikan antara fleksibilitas dari SGML dan kesederhanaan dari HTML.

XML tidak mendefinisikan elemen markup apapun, akan tetapi lebih pada bagaimana membuat elemen sendiri. Dalam artian yang lainnya, XML memberikan kebebasan untuk menentukan sendiri elemen yang akan dibuat. Sehingga elemen dapat menjadi <judul>, <isi> maupun <paragraf>, bergantung pada pengguna. XML tidak memberikan batasan terhadap jumlah elemen yang ingin dibuat. Sejalan dengan kebebasan dalam penulisan XML, ada beberapa aturan yang harus diikuti agar dokumen memiliki format sesuai dengan standar dan mampu diterjemahkan oleh aplikasi penerjemah XML.

XML menjadi lebih sulit ketika hadir dalam bentuk struktur. Sebuah dokumen semestinya tidak memiliki kerancuan dalam pengartian nama, urutan dan struktur dari elemen. Hal ini secara drastis mengurangi tingkat kesalahan dan kompleksitas kode. Aplikasi tidak harus menebak atau mencoba untuk

memperbaiki sintaks yang salah sebaimana yang sering dilakukan oleh *browser* HTML, sehingga tidak heran apabila satu aplikasi pemroses XML menerjemahkan dengan arti yang berbeda. Hal ini membuat penulisan XML menjadi lebih sulit. Seringkan dilakukan pemeriksaan terhadap sintaks melalui sebuah penerjemah untuk memastikan bahwa dokumen yang dibuat akan berjalan dengan kesalahan yang minimal.

Sebagai tambahan dalam pemeriksaan sintaks dasar, dapat dibuat satu aturan bagaimana seharusnya dokumen terlihat. DTD (*Document Type Declaration*) adalah satu kerangka dalam struktur dokumen. Sebuah skema XML dapat pula membatasi tipe data yang diperbolehkan dalam elemen (misalkan, tanggal, angka atau nama). Kemungkinan terhadap pemeriksaan kesalahan dan pengendalian struktur adalah elemen yang digunakan (misalkan, tanggal, angka atau nama).

Untuk penyajian agar memiliki fleksibilitas yang tinggi, dapat digunakan dokumen perubahan format yang disimpan secara terpisah. XML memperbolehkan penggunaan dokumen format (*stylesheet*) yang mengandung informasi perubahan format. Hal ini memiliki keuntungan antara lain:

- Format dapat digunakan untuk berbagai macam dokumen.
- Jika terjadi perubahan format, maka yang harus diubah adalah dokumen format saja, maka semua dokumen akan ikut berubah.
- Penggunaan dokumen format dapat digunakan untuk keperluan yang berbeda, misalkan untuk pencetakan dan yang lainnya untuk penyajian dalam *web*.
- Isi dari dokumen dan struktur tidak berpengaruh meskipun dilakukan perubahan dalam penyajian data.
- Isi dokumen tidak bercampur dengan istilah format dokumen (misalkan, font, spacing, color, dan sebagainya) sehingga nampak lebih mudah untuk dibaca.
- Tanpa adanya elemen format dalam isi dokumen, dapat dilakukan pemilihan nama yang secara tepat mengartikan kegunaan dari suatu hal. Hal ini mempermudah perbaikan dan perubahan.

XML memberikan keleluasaan dalam perubahan struktur, kemudahan pembacaan dan nampak elegan. Adanya kebebasan untuk menentukan bahasa markup sendiri yang memiliki batasan-batasan logis sesuai dengan aturan XML.

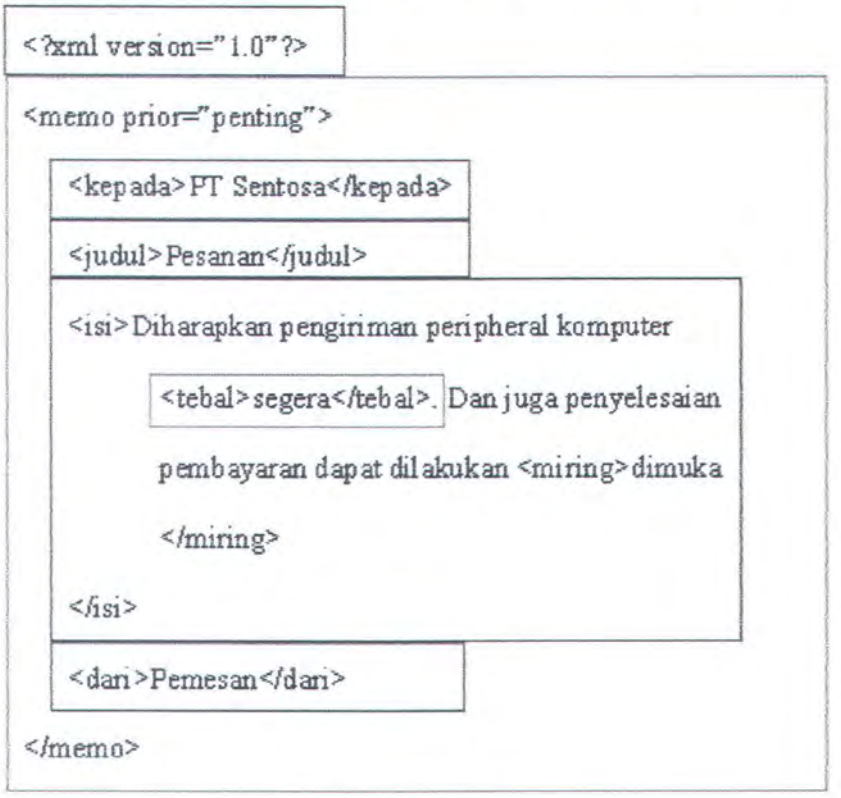
Beberapa bahasa markup cukup lunak mengenai sintaks yang salah dan tidak ditemukan. Ketika kesalahan dibuat dalam sebuah dokumen, hal itu akan membuat penyajian dokumen tidak sesuai dengan yang diharapkan. Kemungkinan penyajian dalam *web browser* yang tidak dapat ditebak, kehilangan informasi, dan program akan bertindak aneh dan kemungkinan gagal ketika mencoba membuka dokumen. XML memberikan batasan-batasan dalam pembuatan dokumen, untuk mengurangi kesalahan yang mungkin dibuat.

Gambar 2.1 merupakan satu contoh XML yang cukup sederhana akan tetapi sangat diterima oleh XML. Contoh ini mengandung isi dengan simbol dari bahasa *markup*. Kurung sudut (< >) dan nama yang berada didalamnya disebut sebagai tag. Tag menentukan dan menandai bagian dari dokumen dan menambahkan informasi yang berguna untuk mendefinisikan struktur. Teks diantara tag merupakan isi dari dokumen, informasi mentah yang kemungkinan adalah isi dari pesan, judul ataupun data lainnya. Markup dan isi melengkapi satu dengan yang lainnya, menghasilkan entitas informasi dengan pembagian, serta penandaan data dalam bentuk yang lebih sederhana.

```
<?xml version=1.0?>
<memo prior=penting>
   <kepada>PT Sentosa</kepada>
   <judul>Pesanan</judul>
   <isi>Diharapkan pengiriman peripheral komputer
      <tebal>segera</tebal>. Dan juga penyelesaian
      pembayaran dapat dilakukan <miring>dimuka
      </miring>
   </isi>
   <dari>Pemesan</dari>
</memo>
```

**Gambar 2. 1 Dokumen XML Sederhana**

Gambar 2.2, menjelaskan tentang pembagian tag dalam dokumen memo, yang diwakili oleh bingkai yang mengelilingi dokumen. Bingkai pertama berisi deklarasi pembuka khusus yang mendukung informasi administratif tentang dokumen tersebut. Bingkai yang lain disebut sebagai elemen. Perilakunya adalah sebagai pembungkus dan penanda dari teks. Elemen terbesar, diberi tanda <memo>, yang melingkupi elemen lainnya dan bertindak sebagai satu paket yang menaungi semua bagian. Didalamnya terdapat elemen khusus yang mewakili fungsi tersendiri dalam dokumen. Pada gambar 2.2, dapat dilihat bahwa elemen terbesar adalah <memo> dan tujuannya (<kepada>), pengirimnya (<dari>), judul pesan (<judul>) dan isi pesan (<isi>). Elemen terakhir adalah yang paling kompleks, campuran elemen dan teks menjadi satu isi. Sehingga dapat dilihat dari contoh ini bahwa sebuah dokumen XML sederhana dapat berisi beberapa tingkatan struktur.




**Gambar 2. 2 Elemen Dalam Dokumen Memo**

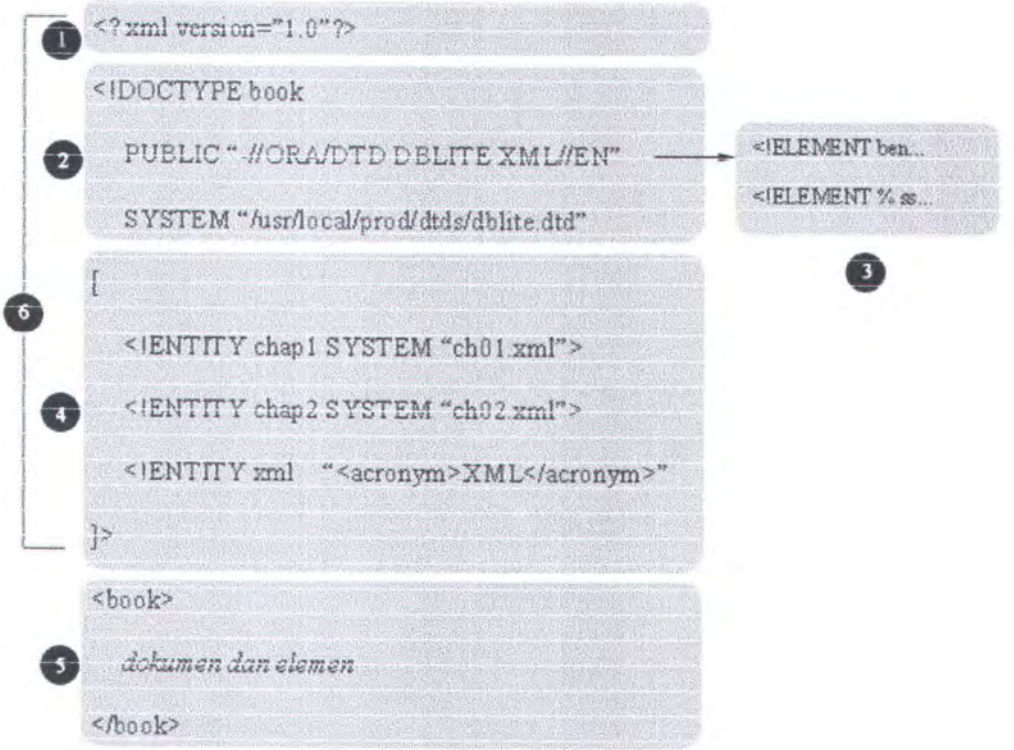
Pada sisi paling atas dari dokumen XML berisi informasi khusus yang dikenal sebagai dokumen pembuka (*document prolog*). Pada susunan paling

sederhana mengatakan bahwa dokumen ini adalah XML dan mendeklarasikan versi XML yang digunakan.

```
<?xml version=1.0?>
```

Pembuka dapat menaungi informasi tambahan seperti detil dari definisi tipe dokumen yang sedang digunakan, deklarasi bagian khusus dari teks, pengkodean teks dan instruksi dari pengolah XML.

Gambar 2.3 menjelaskan mengenai struktur dokumen XML. Pada sisi atas (1) merupakan deklarasi XML. Selanjutnya (2) adalah DTD (*Document Type Declaration*) yang terhubung pada satu dokumen definisi tipe (3) pada dokumen yang berbeda. Kemudian diikuti oleh satu set deklarasi (4). Empat bagian ini tergabung dalam satu prolog (6). Urutan ini tidak dapat diubah, jika terdapat deklarasi XML, maka harus berada pada baris pertama. Jika terdapat DTD, maka harus mendahului elemen *root*.




**Gambar 2. 3 Dokumen Dengan Elemen Prolog dan *Root***

Elemen merupakan bagian dari dokumen. Dokumen dapat dipisahkan menjadi bagian-bagian yang dapat diproses berlainan atau digunakan oleh mesin

pencari. Elemen dapat merupakan satu paket yang bercampur antara teks dengan elemen yang lain. Elemen berikut hanya berisi teks saja:

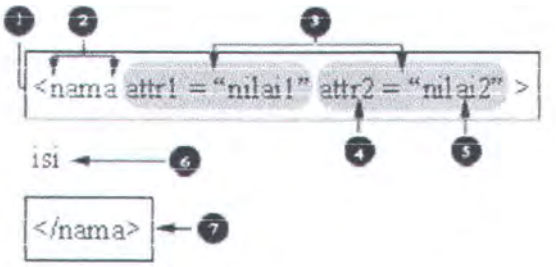
```
<elemen>Ini merupakan teks saja</elemen>
```

Dan elemen berikut berisi teks dan elemen lain:

```
<luar>ini teks <dalam>lain</dalam> dan masih berupa teks
</luar>
```

Beberapa elemen adalah kosong dan memberikan informasi mengenai posisi dan atribut yang dimiliki. Contohnya adalah sebagai berikut:

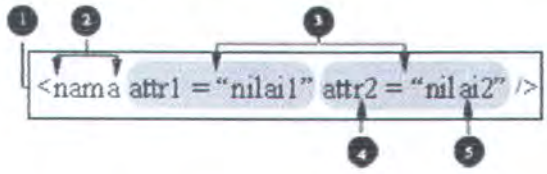
```
<luar>sebuah elemen dapat kosong <kosong//></luar>
```




**Gambar 2. 4 Sintaks Elemen Pembungkus**

Gambar 2.4 menunjukkan bahwa sintaks untuk sebuah elemen pembungkus. Elemen tersebut dimulai dengan tag pembuka (1) yang mengandung sebuah kurung sudut (<) yang diikuti oleh sebuah nama (2). Tag pembuka mungkin mengandung beberapa atribut (3) yang dipisahkan oleh spasi, dan diakhiri dengan kurung sudut penutup (>). Sebuah atribut mendefinisikan anggota elemen dan mengandung sebuah nama (4) yang digabungkan dengan tanda sama dengan (=) untuk sebuah nilai dalam tanda petik (5). Sebuah selemen dapat memiliki beberapa atribut, akan tetapi tidak boleh memiliki nama yang sama. Setelah tag pembuka adalah isi dari elemen (6) yang diikuti oleh tag penutup. Tag penutup terdiri dari kurung sudut pembuka, garis miring, nama elemen dan kurung sudut penutup. Tag penutup tidak memeiliki atribut, dan nama elemen harus sama dengan tag pembuka.

Pada gambar 2.5, sebuah elemen kosong (tidak memiliki isi) mengandung satu tag (1) yang dimulai dengan sebuah kurung sudut pembuka (<) diikuti oleh nama elemen (2). Selanjutnya akan diikuti oleh beberapa macam atribut (3), dimana setiap atribut mengandung nama (4) dan nilai dalam tanda petik (5). Diakhir elemen diakhir dengan garis miring dan sebuah kurung sudut penutup.




**Gambar 2. 5 Sintaks Elemen Kosong**

Ada beberapa aturan yang harus diikuti dalam penulisan elemen dalam XML, yaitu:

- Setiap nama elemen harus diawali dengan huruf atau garis bawah (_) dan dapat diikuti oleh huruf, angka, tanda hubung, garis bawah, titik.
- Nama elemen dapat berupa karakter Romawi, Cyrillic, Yunani, Arab, Thailand, Hiragana, Katakana, Devanagari. Serta ideogram dari Cina, Jepang dan Korea.
- Spasi, tabulasi, baris baru, sama dengan dan karakter petikan yang lain merupakan pemisah antara nama elemen, nama atribut dan nilai atribut, sehingga tidak boleh dijadikan nama elemen.
- Nama elemen dalam XML adalah *case-sensitive*.
- Dalam XML, elemen pembungkus selalu menggunakan tag pembuka dan penutup.
- Elemen XML kosong membutuhkan garis miring sebelum kurung penutup (misalkan, `<misal />`).
- Penulisan elemen pembuka dan penutup harus berurutan.

### 1.7.1 XML *Schema*

Skema adalah cara formal untuk mendefinisikan dan melakukan validasi dari isi sebuah dokumen XML. Dokumen XML dengan bentuk yang benar akan mengkonfirmasi bahwa dokumen tersebut adalah benar. Skema adalah bagaimana

tipe data diberikan pada setiap elemen dan atribut yang terdapat dalam dokumen XML. Skema merupakan dokumen terstruktur yang harus mematuhi aturan sintaks XML. Terdiri dari beberapa bentuk elemen yang sudah didefinisikan sebelumnya dan atribut yang merupakan bagian dari bahasa XML dan digunakan untuk memberikan nilai yang sesuai dengan tipe data dalam elemen tertentu. Skema dapat berupa bagian dari dokumen XML atau terpisah dalam dokumen yang lain.

Aturan XML untuk elemen dan atribut untuk menghasilkan skema adalah sebagai berikut:

- Elemen `Schema` bertindak sebagai pembungkus elemen yang membatasi awal dan akhir dari skema. Elemen ini harus memiliki penutup.
- Atribut `xmlns` digunakan untuk mendeklarasikan tipe dari dari *namespace* skema XML. Nilainya adalah sebuah alamat URL (*Uniform Resource Locator*) atau URN (*Uniform Resource Name*) yang akan digunakan oleh browser untuk mengambil informasi dari tipe data yang diperbolehkan untuk melakukan validasi kode.
- Atribut `xmlns:dt` digunakan untuk mendeklarasikan tipe dari dari namespace skema XML. Nilainya adalah alamat URL atau URN yang akan digunakan oleh *browser* untuk mengambil informasi dari tipe data yang diperbolehkan untuk melakukan validasi kode. Jika digunakan Internet Explorer versi 5 untuk menampilkan dokumen XML, maka harus ditambahkan atribut `xmlns` dan `xmlns:dt` seperti pada gambar 2.6.

```
<Schema xmlns=urn:schema-microsoft-com:xml-data
   xmlns:dt=urn:schemas-microsoft-com:datatypes>
...
```

**Gambar 2. 6 Skema XML yang Ditambahkan untuk Internet Explorer**

- Elemen `AttributeType` digunakan untuk mendeklarasikan tipe data dari sebuah atribut dari sebuah elemen XML. Elemen ini harus diakhiri dengan penutup.
- Attribut `name` untuk penamaan dari atribut.
- Atribut `dt:type` menjelaskan tipe data dari atribut.
- Elemen `attribute` digunakan untuk mengasosiasikan tipe data sebelumnya pada sebuah elemen. Elemen ini harus diberi penutup.
- Atribut `type` untuk tipe data dari atribut tertentu.
- Elemen `ElementType` digunakan untuk mendeklarasikan tipe data dari elemen XML tertentu. Elemen ini harus memiliki penutup.
- Atribut `content` menjelaskan isi dari elemen XML.

### 1.7.2 XML *Link*

*Link* digunakan untuk menghubungkan *web* dengan media untuk meningkatkan nilai dari dokumen, sebagaimana yang dapat dilihat pada gambar 2.7. *Link* pada gambar dikenal sebagai *link* sederhana (*simple links*) karena *link* tersebut hanya menghubungkan dua macam media, setidaknya salah satunya adalah dokumen XML dan bersifat satu arah. Informasi pada *link* ini terdapat didalam satu elemen XML yang bertindak sebagai sisi *link*. Contoh dibawah ini memperlihatkan tentang melakukan *link* pada grafis dan menghubungkan dua dokumen XML secara bersamaan, ini disebut sebagai *simple links*.

**Gambar 2. 7 Hubungan Antar Media**

*Link* yang lebih kompleks dapat mengkombinasikan berbagai macam media dan informasi *link* mungkin disimpan pada lokasi yang tidak berhubungan dengan dokumen sesungguhnya yang akan dihubungkan. Pembahasan yang dilakukan adalah seputar *link* sederhana saja. Beberapa hal yang dapat dilakukan oleh *link* sederhana, yaitu:

- Membagi dokumen menjadi beberapa dokumen dan menggunakan *link* untuk menghubungkan dokumen tersebut.
- Menyediakan navigasi antara komponen dokumen dengan menggunakan *link* untuk membuat menu dari tujuan, daftar isi atau indeks.
- Membuat catatan pada dokumen yang lain dalam internet dengan menyediakan *link* untuk mengambil dan menampilkan data tersebut.
- Mengambil data atau teks dan menampilkannya dalam dokumen dengan menggunakan *link* untuk memasukkan gambar, keluaran program atau hasil dari dokumen yang lain.
- Menyediakan media presentasi, dalam media tersebut dapat dihubungkan dengan film atau potongan musik untuk digabungkan dalam presentasi.
- Menimbulkan aksi secara langsung pada sistem pengguna, misalkan membuka pesan *email*, membaca berita atau membuka kanal media.

Gambar 2.8 merupakan penggalan XML yang menjelaskan bagaimana XLink melakukan pengambilan gambar ke sebuah dokumen. XLink dideklarasikan dalam elemen, yang terletak pada lokasi gambar tersebut akan ditampilkan. Atribut pertama mendefinisikan `namespace` yang disebut `xlink` yang akan digunakan sebagai prefiks untuk semua atribut khusus yang akan menjelaskan *link* tersebut. Atribut berikutnya, `xlink:type`, mendefinisikan *link* dengan tipe *simple*. Tanpa atribut tersebut atribut sisanya tidak akan dijalankan dengan baik. Selanjutnya, atribut `xlink:href` menyimpan URL yang berisi alamat dokumen. Dan paling akhir adalah atribut `xlink:show` yang menspesifikasikan bagaimana *link* tersebut akan diproses. Dalam contoh ini, dokumen akan dipanggil secara langsung dan isinya akan ditampilkan pada posisi

ini dalam dokumen. Dan juga memberitahukan bahwa elemen *link* bersangkutan tidak memeiliki isi, sehingga pengguna tidak harus mengambil secara manual dokumen yang dibutuhkan.

```
<image
    xmlns:link=http://www.w3.org/1999/xlink
    xlink:type=simple
    xlink:href=figs/monkey.gif
    xlink:show=embed
/>
```

**Gambar 2. 8 Pengambilan Gambar Menggunakan XLink**

### 1.7.3 XML *Pointer*

XPointer merupakan tambahan khusus pada sebuah URL yang memerbolehkan untuk mencapai nilai terdalam dari dokumen XML. Untuk lebih mengerti tentang cara kerja dari XPointer maka sebagai contoh dapat dilihat pada *fragment identifier*. *Fragment identifier* adalah mekanisme yang digunakan oleh *link* HTML untuk menghubungkan pada titik tertentu dalam dokumen HTML. Dimana fungsi tersebut menghubungkan pada akhir dari sebuah URL dan dipisahkan dalam URL menggunakan tanda kres (#).

```
<a href=http://www.webku.org/artikel.html#instalasi_linux>
```

Dalam contoh ini, <a> digunakan untuk menghubungkan elemen. Kata-kata yang berada di kanan tanda kres, melanjutkan URL untuk menunjuk pada titik tertentu pada lokasi dalam dokumen `artikel.html`. *Link* mencari sasarannya jika dokumen mengandung penanda dalam bentuk:

```
<a name=instalasi_linux>
```

Persamaan XML dari *fragment identifier* adalah sebuah XPointer, yang diturunkan namanya dari rekomendasi W3C untuk menghubungkan URL dalam *link* XML. Seperti halnya *fragment identifier*, XPointer digabungkan pada sisi kanan dari URL menggunakan tanda kres (#).

```
url#xpointer
```

Dalam contoh yang sangat sederhana, XPointer bekerja sebagaimana *fragment identifier*, menghubungkan ke sebuah elemen pada titik tertentu dengan atribut pengenal. Bagaimanapun juga, XPointer lebih fleksibel karena tujuan dapat berupa elemen apapun. Rekomendasi dari XPointer mendefinisikan keseluruhan bahasa untuk penentuan elemen apapun dalam dokumen meskupun elemen tersebut memeiliki pengenal maupun tidak. Bahasa ini diturunkan dari XPath, sebuah spesifikasi umum untuk mendeskripsikan lokasi di dalam dokumen XML yang didesain untuk memenuhi kebutuhan dari sintaks URL. Pada gambar 2.9 terdapat penggalan kode XML untuk menjelaskan tentang penggunaan XPointer untuk melakukan pencarian terhadap tujuan tertentu dalam dokumen XML. *Sales XPointer* adalah perpendekan dari bentuk `id(sales).id` yang merupakan istilah khusus yang dapat melompat dalam sebuah dokumen pada elemen yang memiliki atribut pengenal yang sama dengan kata dalam tanda kurung. Dapat juga melakukan perpindahan di dalam elemen, misalkan pada elemen *name* dengan isi *Eddie Puss*, maka perpindahan dapat dilakukan dengan menggunakan `id(salesman).child(1,name)`.

```
<?xml version=1.0?>
<personnel>
   <department id=sales>
      <employee>
         <name>Sarah Bellum</name>
         <title>Vice President</title>
         <staff>
            <employee>
               <name>Luke Bizzy</name>
               <title>Manager</title>
               <staff>
                  <employee id=salesman>
                     <name>Eddie Puss</name>
                     <title>Sales Clerk</title>
                  </employee>
               </staff>
            </employee>
         </staff>
      </employee>
   </department>
</personnel>
```

**Gambar 2. 9 Dokumen XML yang Menggunakan XPointer**

### 1.7.4 XSL (*Extensible Stylesheet Language*)

XSL merupakan bahasa perubahan format yang menggunakan aturan tertentu untuk menghasilkan cetakan untuk menampilkan data yang mengandung dokumen XML dalam berbagai cara. XSL terdiri dari dua macam yaitu XSLT (*Extensible Stylesheet Language for Transformation*) dan XSLFO (*Extensible Stylesheet Language Formatting*). XSLT yang akan dijelaskan pada bagian selanjutnya digunakan untuk melakukan transformasi dari dokumen XML ke format lainnya, misalkan HTML, PDF maupun LaTEX. Sedangkan XSLFO merupakan persamaan XML untuk CSS (*Cascading Style Sheet*). XSLFO digunakan untuk mengubah informasi dalam bentuk visual.

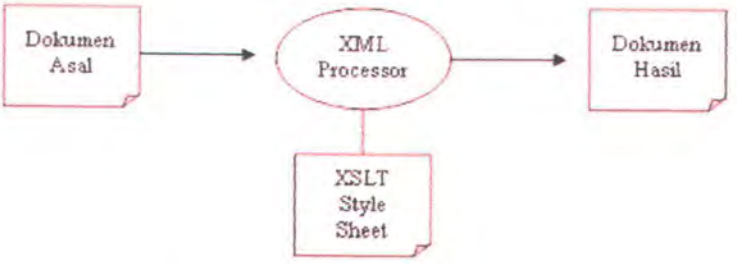
XSL merupakan bahasa yang sederhana. Aturan-aturan yang digunakan dalam XSL sama seperti aturan penulisan XML, misalkan harus memiliki tag

pembuka dan tag penutup. Semua tag dalam XSL memiliki nama awalan yang sama, `xsl:`, untuk menentukan bahwa ini adalah sebuah elemen XSL. Untuk menyajikan dokumen XML, hanya diperlukan tiga elemen XSL, yaitu:

- `xsl:template`, adalah elemen yang digunakan untuk mendefinisikan format dasar. Juga dapat digunakan sebagai elemen pembungkus untuk mendeklarasikan awal dan akhir sebuah perulangan kode XSL.
- `xsl:for-each`, adalah elemen yang digunakan untuk membuat perulangan `for..each` yang memperbolehkan perulangan terhadap semua nilai dari data XML. Atribut `select` digunakan untuk menentukan nama dari elemen data XML.
- `xsl:value-of`, adalah elemen yang digunakan untuk memasukkan nilai dari data XML kedalam format dasar. Atribut `select` digunakan untuk menentukan nama dari data XML. Elemen `xsl:value-of` memerbolehkan penampilan nilai data dari sebuah tag XML.

### 1.7.5 XSLT (*XSL for Transformation*)

XSLT yang akan dijelaskan pada bagian selanjutnya digunakan untuk melakukan transformasi dari dokumen XML ke format lainnya, misalkan HTML, PDF, LaTEX maupun dokumen yang lain. Proses perubahan tersebut dapat diilustrasikan pada gambar 2.10.




**Gambar 2. 10 Penggunaan XSL untuk Mengubah Dokumen**

XSLT tidak dibatasi pada aktivitas pemformatan. Banyak aplikasi yang membutuhkan perubahan format dokumen. Beberapa hal yang dapat dilakukan oleh XSLT adalah sebagai berikut:

- menambahkan elemen khusus untuk menampilkan gambar atau alamat pengirim pada sebuah dokumen XML,
- membuat isi baru dari yang sudah ada, misalkan daftar isi,
- melakukan perubahan dari dokumen ke HTML yang disesuaikan dengan kompatibilitas browser, dan sebagainya.

Sebuah dokumen XSLT memuat aturan pemformatan. Aturan pemformatan memiliki pola yang menentukan elemen yang sesuai dan cetakan yang akan dibuat serta hasil dari pola yang sesuai. Ketika XSLT *processor* mengubah sebuah dokumen XML dibawah kendali dari XSLT *stylesheet*, proses akan berjalan dari *root* dokumen XML dan mengikuti perintah yang sudah didefinisikan dalam aturan pemformatan. *Processor* akan melakukan pemeriksaan terhadap setiap titik dalam dokumen XML, dan akan membandingkan elemen tersebut dengan pola pada cetakan dalam *stylesheet*. Ketika menemukan elemen yang sesuai dengan pola yang dibuat, akan dihasilkan keluaran dari cetakan yang sesuai. Cetakan ini umumnya termasuk juga beberapa markup, data baru, data yang dikopi dari dokumen XML sesuai dengan aturan yang sudah ditetapkan untuk setiap elemen yang akan diproses.

Gambar 2.11 merupakan contoh dokumen XSLT untuk perubahan format dokumen XML kedalam bentuk dokumen HTML.

Ada tiga cara utama untuk melakukan transformasi dari dokumen XML menjadi dokumen lain menggunakan XSLT, yaitu:

- Dokumen XML dan *stylesheet* yang berhubungan dihadirkan bersamaan pada *client* (*web browser*), kemudian dilakukan perubahan format dari dokumen sesuai dengan aturan yang sudah dibuat dan disajikan pada pengguna.
- *Server* memberikan XSLT *stylesheet* pada sebuah dokumen XML untuk mentransformasikannya dalam format lainnya dan mengirimkan hasil transformasi ke *client* (*web browser*).
- Program pendukung lainnya yang mengubah dokumen XML asli menjadi format lain sebelum dokumen diletakkan di *server*. *Server* dan

*client* hanya mengetahui tentang dokumen yang sudah ditransformasikan.

```
<?xml version=1.0?>
<xsl:stylesheet version=1.0
   xmlns:xsl=http://www.w3,org/1999/XSL/Transform>
<xsl:template match=/>
<HTML><BODY>
<xsl:for-each select=library/book>
<DIV>
Judul: <xsl:value-of select=title />
<BR/>
Pengarang: <xsl:value-of select=author />
</DIV>
</xsl:for-each>
</BODY></HTML>
</xsl:template>
</xsl:stylesheet>
```

**Gambar 2. 11 Contoh Dokumen XSLT**

### 1.7.6 GML (*Geographic Markup Language*)

GML merupakan sebuah XML yang berbasis pada pengkodean dengan standar dari informasi geografis yang dibangun oleh OGC (*OpenGIS Consortium*). Status terakhir adalah sebuah RFC (*Request For Comment*) dibawah pengawasan dari OGC. RFC didukung oleh berbagai macam *vendor* termasuk didalamnya Oracle, Galdos System, MapInfo, CubeWerx dan Compusult.

GML merupakan bahasa yang diturunkan dari XML, yang dibangun untuk membantu memisahkan isi dari penyajian, GML melakukan hal ini dalam dunia geografis. GML berkonsentrasi pada representasi dari isi data geografis. GML dapat juga digunakan membuat peta. Hal ini dapat diselesaikan dengan membangun alat bantu untuk mengartikan data GML. Akan tetapi hal ini bertentangan dengan tujuan dari standarisasi dan pemisahan antara sisi dan penyajian. Untuk membuat peta dari GML, dibutuhkan mengubah format dari elemen GML menjadi format yang dapat diterjemahkan oleh tampilan grafis

dalam *web browser*. Tampilan grafis yang mungkin dipakai adalah SVG (*Scalable Vector Graphics*), VML (*Microsoft Vector Language*) dan X3D. Format peta digunakan untuk mengalokasikan elemen GML dan menginterpretasikannya menggunakan format grafis umum.

Seperti halnya pengkodean XML, GML merepresentasikan informasi geografis dalam bentuk teks. Teks memiliki kesederhanaan dan kemudahan pembacaan pada satu sisi. Hal ini memudahkan pemeriksaan dan perubahan yang akan dilakukan.

GML berbasis pada model abstrak geografis yang dibangun oleh OGC. Hal ini mendeskripsikan tentang istilah entitas geografis yang dikenal sebagai fitur/*features*. Fitur adalah tidak lebih dari sebuah daftar properti dan geomerti. Properti memiliki nama umum, tipe, deskripsi nilai. Geometri merupakan komposisi dari geometri bangunan dasar misalkan, titik, garis, kurva, permukaan dan poligon.

Pengkodean GML juga dapat berupa fitur yang cukup kompleks. Sebuah fitur sebagai contoh dapat berupa komposisi dari fitur yang lainnya. Sebuah fitur tunggal seperti bandara udara dapat berupa komposisi dari fitur yang lain, misalkan jalur taksi, jalur terbang, hangar, dan terminal udara. Fitur geografis dari geometri juga dapat berupa komposisi dari berbagai macam elemen geometri. Fitur geometri kompleks dapat terdiri dari campuran dari tipe geometri termasuk titik, garis dan poligon. Gambar 2.12 merupakan penggalan kode GML sederhana untuk gedung sekolah.

Pada rekomendasi XML 1.0 dari W3C, versi terakhir dari GML adalah berdasarkan XML 1.0 dan menggunakan `FeatureCollection` sebagai dasar dari dokumen yang dibuat. `FeatureCollection` adalah koleksi dari fitur GML dengan `Envelope` (merupakan batasan dari set fitur), kumpulan properti yang digunakan dalam `FeatureCollection` dan daftar pilihan dari *Spatial Reference System Definitions*. Sebuah `FeatureCollection` dapat juga berisi `FeatureCollection` yang lain, yang menunjang `Envelope` dari pembatasan `FeatureCollection` yang membatasi `Envelope` dari semua yang mengandung `FeatureCollection`.

```
<Feature   fid=142 featureType=school  Description=A middle
school>
   <Polygon name=extent srsName=epsg:27354>
      <LineString name=extent srsName=epsg:27354>
         <CData>
            491888.999999459,5458045.99963358
         491904.999999458,5458044.99963358
            491908.999999462,5458064.99963358
         491924.999999461,5458064.99963358
            491925.999999462,5458079.99963359
         491977.999999466,5458120.9996336
            491953.999999466,5458017.99963357
         </CData>
      </LineString>
   </Polygon>
</Feature>
```

**Gambar 2. 12 Contoh Dokumen GML**

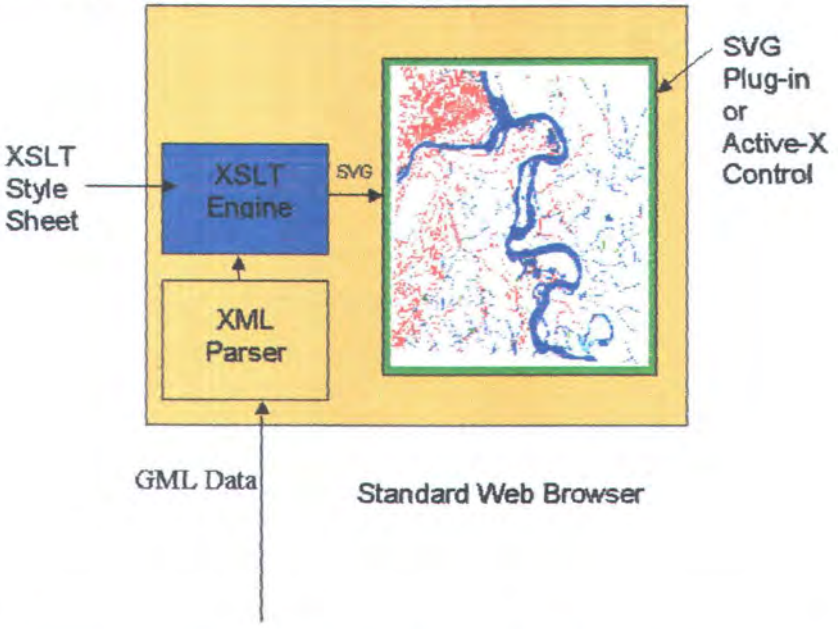
Ketika permintaan dibuat untuk data GML dari *server* GML, data selalu dikembalikan dalam bentuk `FeatureCollection`. Disini tidak ada batasan dalam RFC GML tentang jumlah *feature* yang dapat dimuat dalam `FeatureCollection`. Hal ini dikarenakan `FeatureCollection` dapat berisi `FeatureCollection` lainnya yang masih berhubungan secara sederhana untuk penggabungan `FeatureCollection` yang diterima dari sebuah *server* dalam koleksi yang lebih besar.

Ketika GML secara efektif berarti transportasi informasi geografis dari satu tempat ke tempat lainnya, dapat diharapkan bahwa hal tersebut menjadi hal penting dalam penyimpanan informasi geografis dengan baik. Elemen kuncinya adalah XLink dan XPointer. Sementara dua spesifikasi ini dalam pengembangan dan implementasi, keduanya memiliki peranan penting dalam pembangunan set data geografis kompleks dan terdistribusi. Data geografis secara lazim tersebar diberbagai tempat di dunia. GML dipercaya nantinya digunakan sebagai bentuk penyimpanan yang mengkombinasikan antara XLink dan XPointer yang

memberikan kontribusi yang sangat berguna untuk permasalahan integrasi dan standarisasi peta.

Beberapa spesifikasi dasar XML untuk mendeskripsikan elemen grafis vektor telah dikembangkan, termasuk didalamnya adalah SVG, VML dan X3D. Untuk menampilkan tiga elemen grafis vektor diatas diperlukan aplikasi tambahan. SVG dapat ditampilkan dengan menambahkan aplikasi bantuan yang dikembangkan oleh Adobe untuk *web browser* Internet Explorer dan Netscape Communicator. Beberapa aplikasi SVG berbasis Java juga tersedia. Untuk VML sudah terpaket secara langsung dalam Internet Explorer 5.0.

Untuk menampilkan peta dari GML dibutuhkan pengubahan data dari GML menjadi salah satu dari format data vektor grafis antara lain SVG, VML atau X3D. Hal ini berhubungan dengan format grafis (misalkan simbol, warna, tekstur) dalam setiap tipe fitur GML atau instan fitur. Gambar 2.13 merupakan ilustrasi proses transformasi dokumen GML menjadi SVG.




**Gambar 2. 13 Proses Transformasi GML ke SVG menggunakan XSLT**

GML merupakan pengkodean fitur geografis sederhana berbasis teks. GML didasarkan pada model umum dari geografis (Spesifikasi Abstrak OGC) yang telah dikembankan dan disetujui perkembangannya secara cepat oleh berbagai vendor SIG di berbagai belahan dunia. Hal yang paling penting adalah

GML berbasis pada XML. Hal ini dianggap penting karena XML memberikan kemampuan untuk melakukan verifikasi integritas data. Yang kedua adalah berbagai macam dokumen XML dapat dibaca dan diubah menggunakan editor teks sederhana. Yang ketiga adalah sejak adanya penambahan jumlah dari bahasa XML, akan lebih mudah untuk melakukan integrasi data GML dengan data non spasial. Dan yang paling penting adalah XML mudah untuk ditransformasikan. Dengan menggunakan XSLT atau bahasa pemrograman yang lainnya, transformasi XML dapat secara mudah dilakukan dari satu bentuk ke bentuk yang lain.

### 1.7.7 SVG (*Scalable Vector Graphic*)

SVG merupakan bahasa yang digunakan untuk mendeskripsikan grafis dua dimensi dalam XML. SVG memperbolehkan tiga tipe dari obyek grafis, yaitu bentuk vektor grafis (misalkan jalur yang terdiri dari garis lurus dan kurva), gambar dan teks. Obyek grafis dapat dikelompokkan, diubah formatnya, ditranformasikan dan dikomposisikan menjadi obyek lainnya. Kemampuan yang diberikan termasuk juga transformasi berulang, penempelan *path*, *alpha masks*, efek filter dan cetakan untuk obyek.

Hasil dari SVG dapat juga interaktif dan dinamis. Animasi dapat didefinisikan dan ditimbulkan secara deklaratif (misalkan, dengan menempelkan elemen animasi SVG pada isi SVG) atau dengan menggunakan skripting.

Aplikasi SVG terbaru juga memungkinkan penggunaan dari bahasa skripting tambahan yang melakukan akses pada SVG DOM (*Document Object Modelling*), yang memberikan kemampuan untuk mengakses secara penuh pada semua elemen, atribut dan properti. Sebuah set dari penanganan even seperti *onmouseover* dan *onclick* dapat digunakan pada obyek grafis SVG apapun. Karena kompatibiltas dan peningkatan dari standar *web* lainnya, fitur seperti skripting dapat diterapkan pada XHTML dan elemen SVG secara bersamaan pada halaman *web* yang sama.

SVG menyediakan elemen *path* umum, yang dapat digunakan untuk menghasilkan berbagai macam variasi dari obyek grafis, dan juga menyediakan

bentuk dasar umum seperti bujur sangkar dan elips. Ini merupakan kenyamanan untuk melakukan pengkodean manual dan mungkin digunakan untuk cara yang sama dalam elemen path umum lainnya. SVG memberikan pengendalian kualitas melalui sistem koordinat dari obyek grafis yang telah didefinisikan dan transformasi yang akan digunakan selama proses render.

SVG memungkinkan pengguna untuk membuat, menggunakan kembali dan membagi simbol yang dibuat tanpa memerlukan sentralisasi *registry*. Komunitas pengguna dapat membuat dan meningkatkan kualitas simbol sesuai dengan kebutuhannya tanpa harus meminta persetujuan komite. Desainer dapat secara tepat mengatur penyajian simbol grafis yang digunakan tanpa khawatir akan simbol yang tidak didukung. Simbol mungkin digunakan dalam ukuran dan orientasi yang berbeda dan dapat juga diformat ulang agar dapat digunakan dalam komposisi grafis yang berbeda.

Banyak grafis web yang menggunakan operasi filterisasi yang ditemukan dalam paket pewarnaan untuk membuat efek samar, bayangan, pencahayaan dan sebagainya. Dengan menggunakan rasterisasi *client-side* yang digunakan dengan format vektor hal ini sangatlah tidak mungkin. SVG memperbolehkan spesifikasi deklaratif dari filter, baik digunakan secara tunggal maupun kombinasi, yang dapat digunakan dalam sisi *client* ketika SVG diproses. Hal inilah yang membuat grafis masih dapat ditampilkan pada resolusi yang berbeda.

SVG juga memberikan elemen font baik berupa teks maupun grafis. Hal ini bertujuan untuk mengurangi permasalahan yang sering timbul dalam hal perbedaan format font yang dimiliki oleh client yang berbeda-beda.

Gambar 2.14 merupakan penggalan dokumen SVG yang sudah ditransformasikan oleh XSLT dari dokumen GML untuk data geografis sebuah kelurahan. [6].

```
<?xml version=1.0 encoding=utf-8 ?>
<svg width=300 height=300 onclick=showFeatureData(evt)>
   <script xlink:href=svgmap.js />
   <path id=kelurahan01 fill=red
      d=M100,100 200,200 250,150 225,125 100,100 />
   .
   .
</svg>
```

**Gambar 2. 14 Contoh Dokumen SVG Hasil Transformasi dari Dokumen GML**

## 1.8 JAVA *OPEN SOURCE*

Berikut merupakan *library-library open source* berbasis Java yang digunakan dalam pengerjaan tugas akhir ini.

### 1.8.1 Deegree

Proyek Deegree (*Deegree Project*) merupakan sebuah proyek *open source* yang diimplementasikan menggunakan *Java* sebagai bahasa pengembangannya. Deegree memiliki kemampuan dalam pembangunan infrastruktur data spasial sesuai dengan standar dari *Open GIS Consortium* (OGC) dan ISO/TC 211. Keseluruhan arsitektur deegree didasarkan pada konsep dan spesifikasi OGC dan tidak memiliki permasalahan untuk diintegrasikan dengan produk standar dari *vendor* lain. Layanan berbasis OGC yang dimiliki oleh deegree antara lain:

- WMS (*Web Map Services*), merupakan pembuatan peta berbasis *web*, hasilnya dapat ditampilkan menggunakan *web browser*.
- WFS (*Web Feature Services*), merupakan pengaksesan data vektor geografis berbasis *web* sesuai dengan standar GML 2.1.1 pada pengguna lokal, untuk selebihnya melakukan proses terhadap data yang diambil.
- WCS (*Web Coverage Services*), merupakan pengaksesan data raster geografis berbasis *web* yang dapat disajikan menggunakan beberapa format gambar antara lain TIFF, GIF, JPEG, BMP, dan PNM.

- WCAS (*Web Catalog Services*) berbasis *OGC Web Services Stateless Catalog Profile*, merupakan layanan katalog berbasis *web* untuk administrasi dan pencarian dari *metadata* yang mendeskripsikan data dan layanan geografis. Sebuah layanan katalog memperbolehkan pengambilan data dan layanan berbasis pada kriteria pencarian spasial dan tekstual.
- WFS-G (*Web Gazetteer Services*), merupakan layanan yang mengijinkan pereferensian geografis dari entitas geografis berbasis pada pengenal tekstual, misalkan nama tempat.
- WTS (*Web Terrain Services*), merupakan layanan yang menghasilkan penyajian data tiga dimensi seperti model kota dan elevasi *digital*. Hasil penyajian dapat dihadirkan menggunakan *web browser* standar.
- WCTS (*Web Coordinate Transformation Services*), merupakan layanan yang memberikan fasilitas transformasi dari koordinat geografis dari satu sistem referensi koordinat ke sistem lainnya.

Hingga saat ini deegree telah berhasil dijalankan di Microsoft Windows NT/2000, Linux, Macintosh OS X dan Solaris *Server*. Pembacaan dan penulisan terhadap sumber data yang ada misalkan, *ORACLE spatial*, PostGres/PostGIS, MySQL, *database* lain yang menggunakan JDBC, ESRI *Shapefile*, beberapa format data raster (JPEG, GIF, PNG, (Geo)TIFF, PNM dan BMP) dan layanan lainnya meskipun menggunakan format *propietary*. Data vektor disajikan oleh deegree dalam bentuk GML 2.1.1. Penyajian dalam bentuk ESRI *Shapefile* juga merupakan pilihan yang diberikan.

*Library* dari deegree digunakan dalam pengerjaan tugas akhir ini pada proses konversi dari *shapefile* ke GML.

### 1.8.2 Saxon

Paket saxon merupakan satu set alat bantu untuk melakukan pemrosesan dokumen XML. Komponen yang dimiliki oleh saxon adalah sebagai berikut:

- XSLT *processor* yang mengimplementasikan rekomendasi versi 1.0 untuk XSLT dan *XPath* dari *World Wide Web Consortium* dengan beberapa ekstensi handal.
- Pustaka *Java*, yang mendukung model pemrosesan yang hampir sama seperti XSL, akan tetapi memperbolehkan kapabilitas pemrograman secara penuh yang dibutuhkan untuk melakuan pemrosesan secara kompleks terhadap data atau untuk mengakses layanan luar misalkan *database* relasional.
- Peningkatan improvisasi dari *Aelfred parser* yang secara resmi ditulis oleh David Megginson kemudian oleh Microstar.

Saxon dapat digunakan dalam penulisan XSLT *stylesheet*, penulisan aplikasi *Java* atau kombinasi keduanya. Saxon umumnya berguna ketika melakukan konversi data XML menjadi format lainnya. Format keluaran dapat saja berupa XML, HTML, format lainnya misalkan nilai yang dipisahkan oleh koma, *EDI messages*, atau data dalam *database* relasional.

*Library* dari saxon digunakan dalam pengerjaan tugas akhir ini pada proses konversi dari GML ke SVG.

### 1.8.3 Batik

Batik merupakan teknologi *Java* berbasis alat bantu (*toolkit*) untuk aplikasi ataupun *applet* yang akan menggunakan gambar dalam format SVG untuk berbagai macam keperluan misalkan penyajian, pemrosesan dan manipulasi.

Tujuan dari Batik sendiri adalah memberikan pengembang perangkat lunak satu set modul inti yang dapat digunakan secara bersamaan atau individual untuk mendukung solusi spesifik SVG. Contoh dari modul inti adalah *SVG Parser, SVG Generator*, SVG DOM, dan PNG *Transcoder*.

*Library* dari batik digunakan dalam pengerjaan tugas akhir ini pada proses konversi dari SVG ke *bitmap*.

# BAB III

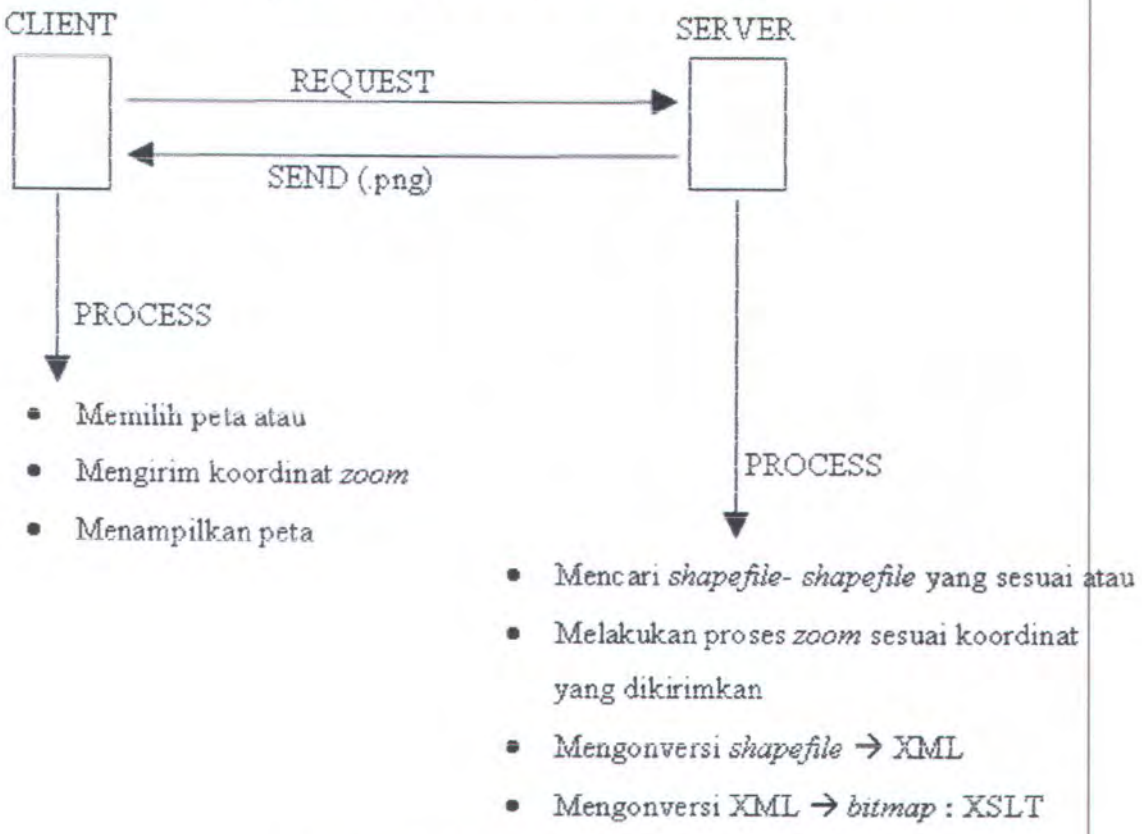
# PERANCANGAN PERANGKAT LUNAK

# BAB III
# PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas perancangan dari sistem visualisasi peta yang meliputi empat aspek. keempat aspek tersebut adalah arsitektur *client-server*, perancangan data, perancangan proses, dan perancangan antar-muka perangkat lunak. Bab ini akan memberikan gambaran dasar mengenai perangkat lunak yang akan dikerjakan.

## 3.1 ARSITEKTUR CLIENT-SERVER

Secara garis besar, arsitektur proses yang terjadi pada sistem ini meliputi sisi *client* dan *server* yang akan dibuat, dapat dilihat pada gambar 3.1.




**Gambar 3. 1 Arsitektur Client-Server**

Dengan penjelasan gambar sebagai berikut :

1. *Client* mengirimkan data peta beserta fitur yang diinginkan (mis: rumah sakit di Kecamatan Taman) ke komputer *server*.
2. Atau *client* mengirimkan koordinat *zoom* pada peta, yang diinginkan.
3. *Server* akan mengolah data peta beserta fitur tersebut dan mencari *shapefile-shapefile* yang sesuai.
4. Atau *server* akan melakukan proses *zoom* pada peta sesuai koordinat yang telah dipilih.
5. *Shapefile* yang sesuai kemudian diubah menjadi format XML.
6. Baru *file* XML tersebut dikonversi ke format .png dengan menggunakan XSLT.
7. *File* gambar dengan format .png tersebut yang kemudian akan dikirim ke *client* untuk ditampilkan.

## 3.2 PERANCANGAN DATA

Tujuan dari tahap ini adalah untuk memersiapkan data yang akan digunakan dalam tahap implementasi perangkat lunak nanti. Perancangan data ini akan dibagi menjadi tiga bagian, yaitu data masukan, data proses, dan data keluaran.

### 3.2.1 Data Masukan

Data masukan yang akan digunakan adalah data Kota Madiun yang berformat *shapefile*. Data ini merupakan *output* dari perangkat lunak ESRI ArcView yang memiliki format *propietary*.

*Shapefile* ESRI menghasilkan tiga dokumen utama agar data tersebut mampu dibaca dan digunakan kembali oleh aplikasi yang sama maupun aplikasi lainnya. Ketiga dokumen yang dihasilkan tersebut memiliki format dan fungsi yang berbeda dan tidak dapat dipisahkan. Maksudnya adalah bahwa ketiganya harus tersedia dan tidak dapat digunakan apabila salah satunya tidak ada. Berikut akan dijelaskan dengan singkat fungsi dari masing-masing dokumen yang membentuk sebuah *Shapefile*, yaitu:

- Dokumen utama, berekstensi .shp. Dokumen ini merupakan dokumen utama yang dapat dibaca oleh aplikasi secara langsung. Berisi urutan data yang menjelaskan bentuk obyek geometri beserta daftar vertex yang membentuknya.
- Dokumen index, berekstensi .shx. Dokumen ini berisi *offset* dari dokumen utama yang tersusun berurutan mulai dari awal dokumen.
- Dokumen data, berekstensi .dbf. Dokumen ini berisi atribut fitur data geometri. Relasi *one-to-one* antara data geometri dan fitur-fiturnya tersusun berurutan sesuai dengan urutan pada dokumen utama. Dokumen ini memiliki format dokumen *database* dBASE.

### 3.2.2 Data Proses

Data proses merupakan data yang diperlukan selama proses konversi *shapefile* ke GML, GML ke SVG, dan dari SVG ke *bitmap*. Data ini digolongkan menjadi dua macam data, yaitu data hasil konversi dengan format GML dan SVG, dan data tambahan untuk melakukan proses transformasi data GML ke SVG berupa data XSLT. Secara detil data tersebut akan dijelaskan sebagai berikut:

- Data konversi, berekstensi .gml. Data ini merupakan pengkodean fitur geografis dalam bentuk dokumen dengan format XML. Data ini disimpan dalam bentuk tag-tag GML.

  Data konversi, berekstensi .svg. Data ini berupa tag-tag SVG hasil transformasi dari dokumen GML menggunakan XSLT untuk memudahkan pembacaan lebih lanjut. Dokumen dengan format SVG ini dibentuk sesuai dengan kerangka penerjemah yang diatur oleh dokumen XSLT. Dokumen SVG merupakan penyajian visual dengan tipe vektor dari fitur geografis dokumen GML.
- Data transformasi, berekstensi .xsl. Data ini digunakan untuk melakukan proses transformasi format data GML menjadi format data SVG yang memiliki kemampuan untuk disajikan secara visual. Data ini memuat aturan pewarnaan dan penyajian dokumen.

### 3.2.3 Data Keluaran

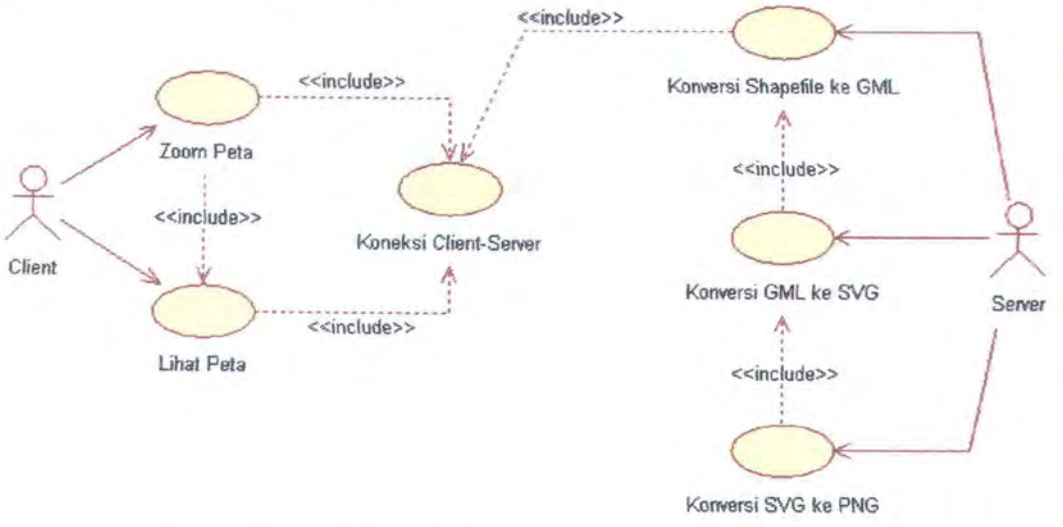
Data keluaran dari perangkat lunak ini adalah *file* gambar dengan format .png. *File* ini dihasilkan dari pengonversian gambar dengan tipe vektor SVG ke gambar dengan tipe *raster bitmap*. Gambar *bitmap* inilah yang nantinya akan dikirim oleh server ke Pengguna untuk dilihat pada ponselnya.

## 3.3 PERANCANGAN PROSES

Pendekatan yang digunakan pada perancangan perangkat lunak ini adalah perancangan berbasis obyek menggunakan UML (*Unified Modelling Language*) dengan perangkat lunak Rational Rose Enterprise Edition 2002. Pemodelan sistem visualisasi peta ini akan dihadirkan dalam bentuk diagram. Pertama akan digunakan diagram *use-case* untuk menggambarkan fungsionalitas yang dimiliki aplikasi. Setelah itu langkah-langkah yang dilakukan pada tiap *use-case* akan dijelaskan lebih lanjut lagi dalam diagram *sequence*. Sedangkan proses konversi, yang merupakan proses utama pada perangkat lunak, akan digambarkan dengan menggunakan diagram aktivitas.

### 3.3.1 Diagram *Use-Case*

Gambar 3.2 merupakan diagram *use-case* untuk sistem visualisasi peta. Diagram ini dibangun dari penentuan entitas yang diperlukan dalam perangkat lunak ini, yaitu aktor dan *use-case*.




**Gambar 3. 2 Diagram *Use-Case* Sistem Visualisasi Peta**

Pada sistem visualisasi peta ini ada dua entitas aktor, yaitu Client dan Server. Pada sisi Client terdapat dua *use-case*, yaitu Lihat Peta dan Zoom. Tetapi untuk melakukan proses Zoom, proses Lihat Peta harus telah dijalankan paling tidak sekali. Kedua proses pada Client tersebut juga harus menjalankan proses Koneksi Client-Server. Pada Server berjalan tiga *use-case*, yaitu Konversi Shapefile ke GML, Konversi GML ke SVG, dan Konversi SVG ke PNG. Secara berurutan, seperti yang dapat dilihat pada gambar 3.2, alur pada server berjalan mulai dari koneksi ke ketiga proses konversi.

### 3.3.2 Diagram *Sequence*

Fungsionalitas dan gambaran dasar yang nantinya akan oleh dimiliki perangkat lunak telah dijelaskan pada diagram *use-case* di atas. Selanjutnya akan diberikan penjabaran lebih lanjut lagi mengenai alur proses dan hubungan antar obyek terkait pada sistem visualisasi peta ini.

#### 3.3.2.1 *Sequence* Lihat Peta

Berikut merupakan rangkaian proses dan obyek yang terlibat pada fungsionalitas Lihat Peta yang telah digambarkan pada diagram *use-case* sebelumnya. Obyek-obyek dan proses-proses tersebut dimodelkan dalam diagram *sequence* seperti yang dapat dilihat pada Gambar 3.3 di bawah. Diagram ini juga berisi penjabaran lebih lanjut dari *use-case* Lihat Peta.

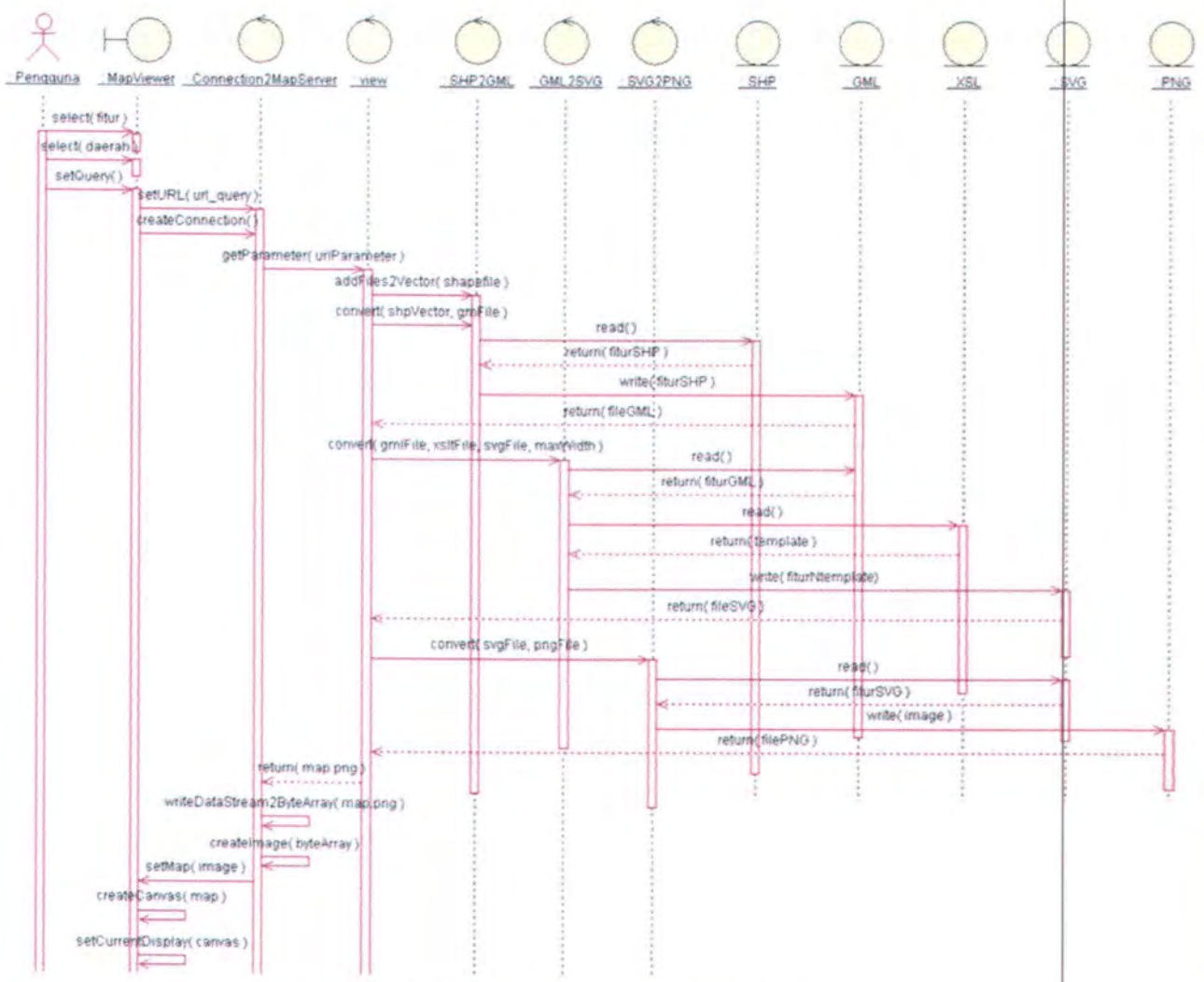
Pada proses Lihat Peta ini yang pertama dilakukan oleh pengguna adalah memilih fitur yang ingin dilihat pada peta nantinya, seperti: Rumah Sakit, Penginapan, dll. Kemudian pengguna memilih area yang ingin ditampilkan, seperti: Kecamatan Manguharjo, Kecamatan Kartoharjo, dll.

Setelah informasi mengenai fitur dan daerah yang dipilih telah dimasukkan oleh pengguna, `Mapviewer` kemudian akan mengolah data tersebut menjadi *query* dan ditambahkan ke url pada `Connection2MapServer` dengan menggunakan `setURL(String url_query)`. Baru koneksi ke server dibuat dengan menggunakan `createConnection()` ke url yang telah di-set.

Data dari pengguna yang dikirimkan kemudian akan diambil oleh `view`. *Shapefile-shapefile* yang diperlukan lalu akan dicari menggunakan data ini tadi. *Shapefile-shapefile* yang telah didapatkan ini kemudian dikonversi menjadi *file* .gml menggunakan *method* `convert(Vector shpVector, String gmlFile)` dari `SHP2GML`. Kemudian oleh `GML2SVG`, *file* .gml yang telah didapat dikonversi lagi menjadi *file* .svg dengan bantuan XSLT menggunakan `convert(String gmlFile, String xsltFile, String svgFile, int maxWidth)`.

`MaxWidth` di sini adalah ukuran lebar layar ponsel pengguna. Variabel ini diperlukan untuk mengatur lebar gambar peta yang dihasilkan agar tidak melebihi lebar layar ponsel pengguna. Hal ini dilakukan karena ponsel tidak dapat melakukan *scrolling* gambar secara horizontal.

*File* .svg yang dihasilkan lalu dikonversi menjadi *file* .png oleh `SVG2PNG` menggunakan `convert(String svgFile, String pngFile)`. *File* .png inilah yang kemudian akan dibaca dan dijadikan `Image` oleh `Connection2MapServer`. `Image` ini oleh `MapViewer` harus terlebih dahulu diletakkan dalam `Canvas` sebelum ditampilkan karena untuk keperluan fungsionalitas *zoom* diperlukan *method* `repaint()` milik `Canvas`. `Canvas` dengan `Image` dari *file* .png inilah yang akan dilihat oleh pengguna pada ponselnya sebagai hasil akhir dari peta yang telah dipilihnya.




**Gambar 3. 3 Diagram *Sequence* Lihat Peta**

### 3.3.2.2 *Sequence Zoom*

Berikut merupakan rangkaian proses dan obyek yang terlibat pada fungsionalitas *Zoom* yang telah digambarkan pada diagram *use-case* sebelumnya. Obyek-obyek dan proses-proses tersebut dimodelkan dalam diagram *sequence* seperti yang dapat dilihat pada Gambar 3.4 di bawah. Diagram ini juga berisi penjabaran lebih lanjut dari *use-case Zoom*.

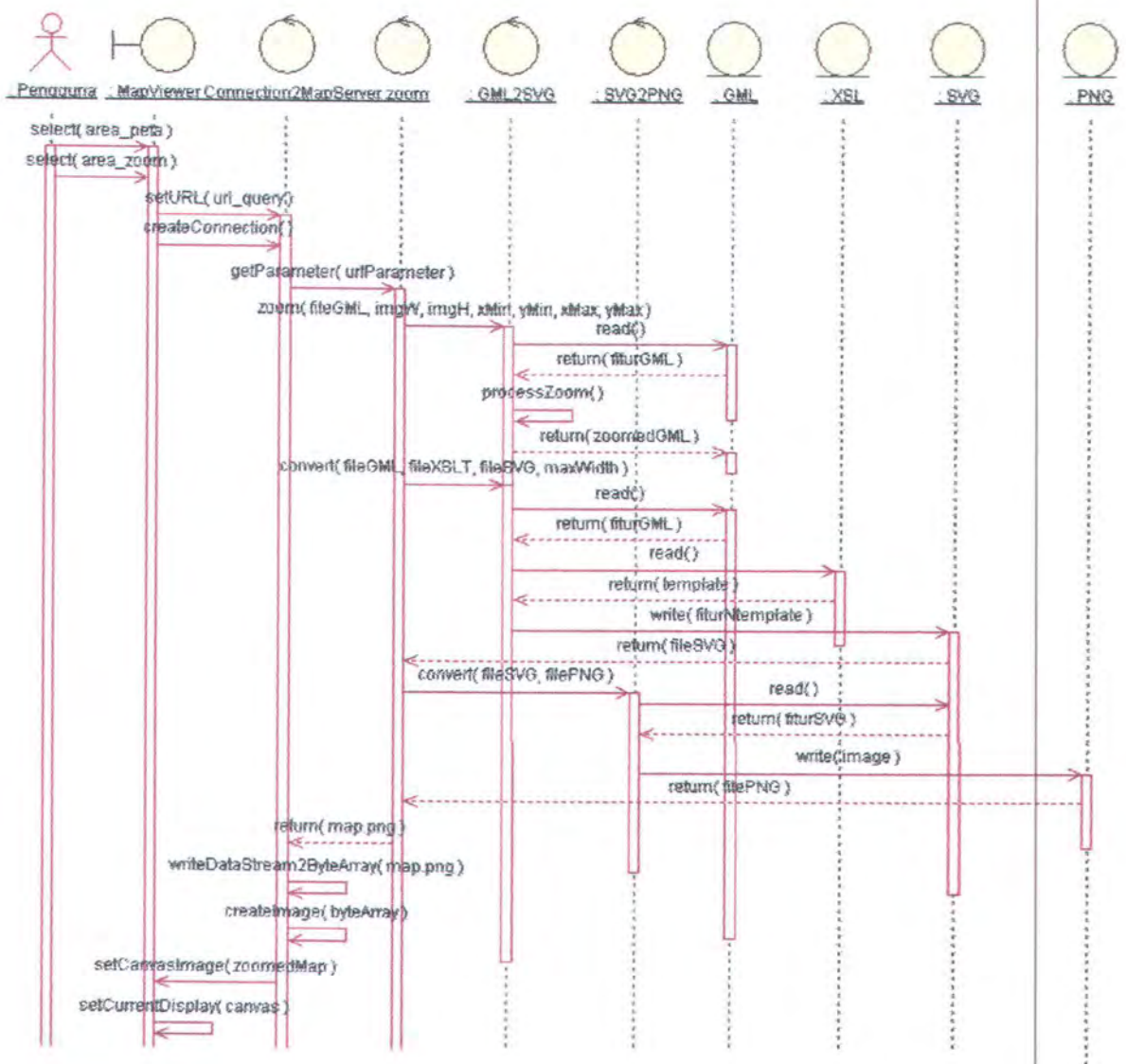
Pada proses *Zoom* ini karena ada kemungkinan peta yang dilihat lebih besar dari layar ponsel, maka yang pertama kali dilakukan oleh aktor pengguna adalah memilih daerah pada peta yang ingin di-*zoom*. Baru kemudian pengguna dapat memilih area pada peta yang ingin dilihat secara lebih dekat.

Setelah data koordinat area *zoom* telah dimasukkan oleh pengguna, data tersebut akan ditambahkan sebagai *query* pada url yang terdapat di `Connection2MapServer`. Baru koneksi ke server dapat dibuat dengan menggunakan `createConnection()` ke url yang telah di-set.

Data dari pengguna dalam bentuk parameter *query* url yang telah dikirimkan kemudian akan diambil oleh `zoom`. Data tadi lalu diolah menjadi variabel-variabel yang diperlukan dalam proses *zoom* ini. Variabel-variabel yang telah didapatkan ini tadi kemudian digunakan sebagai parameter *method* `zoom(String gmlFile, int imageWidth, int imageHeight, int xMin, int yMin, int xMax, int yMax)` dari `GML2SVG` untuk mengubah *file* .gml sesuai dengan koordinat *zoom* yang diminta pengguna. *File* .gml yang telah didapat kemudian dikonversi menjadi *file* .svg dengan bantuan XSLT menggunakan `convert(String gmlFile, String xsltFile, String svgFile, int maxWidth)`.

`MaxWidth` di sini adalah ukuran lebar layar ponsel pengguna. Variabel ini diperlukan untuk mengatur lebar gambar peta yang dihasilkan agar tidak melebihi lebar layar ponsel pengguna. Hal ini dilakukan karena ponsel tidak dapat melakukan *scrolling* gambar secara horizontal.

*File* .svg yang dihasilkan lalu dikonversi menjadi *file* .png oleh `SVG2PNG` menggunakan `convert(String svgFile, String pngFile)`. *File* .png inilah yang kemudian akan dibaca dan dijadikan `Image` oleh `Connection2MapServer`. `Image` ini oleh `MapViewer` lalu diletakkan dalam `Canvas`. `Canvas` dengan `Image` dari *file* .png inilah yang akan dilihat oleh pengguna pada ponselnya sebagai hasil akhir dari peta yang telah dipilihnya.




**Gambar 3. 4 Diagram *Sequence Zoom***

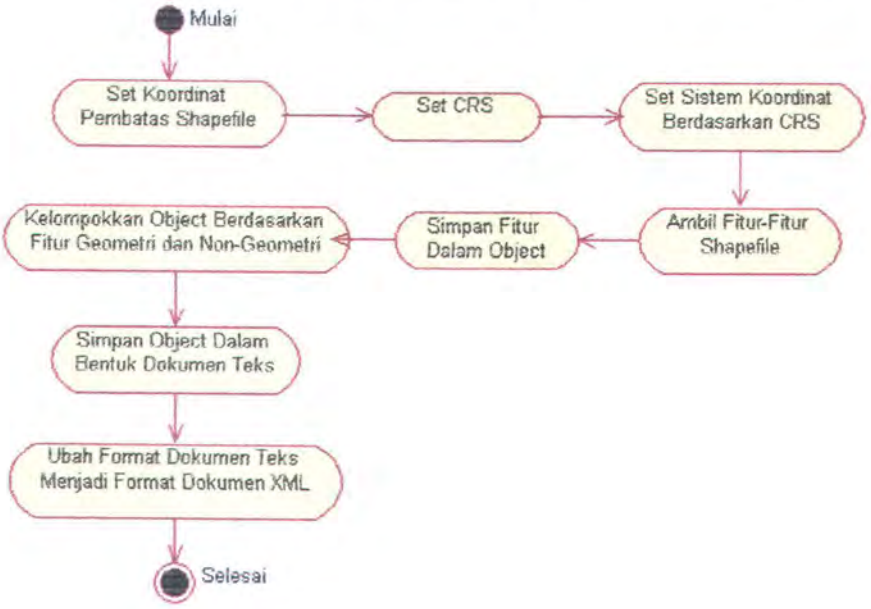
### 3.3.3 Diagram Aktivitas

Pada bagian ini akan diberikan gambaran dasar mengenai alur dan tahapan yang dilalui pada tiap proses konversi.

#### 3.3.3.1 Aktivitas Konversi *Shapefile* ke GML

Pada proses konversi *shapefile* ke format gml ini, seperti dapat dilihat pada gambar 3.5 di bawah, tahap pertama yang harus dilakukan adalah menentukan koordinat-koordinat yang membatasi shapefile. Tentukan juga CRS (*Coordinate Reference System*) untuk *shapefile* berdasarkan garis bujur dan lintang relatif

pada Greenwich. Setelah itu set sistem koordinat transformator berdasarkan CRS yang telah didapat. Kemudian baru fitur-fitur yang terdapat pada *shapefile* diambil. Simpan fitur-fitur tersebut dalam object dan kelompokkan object-object yang berisi fitur dari *shapefile* tadi berdasarkan properti geometri atau non-geometri. Langkah selanjutnya adalah penyimpanan object-object yang telah dikelompokkan dalam bentuk dokumen teks atau String. Terakhir, ubah format dokumen dari teks biasa menjadi dokumen dalam format penulisan XML.



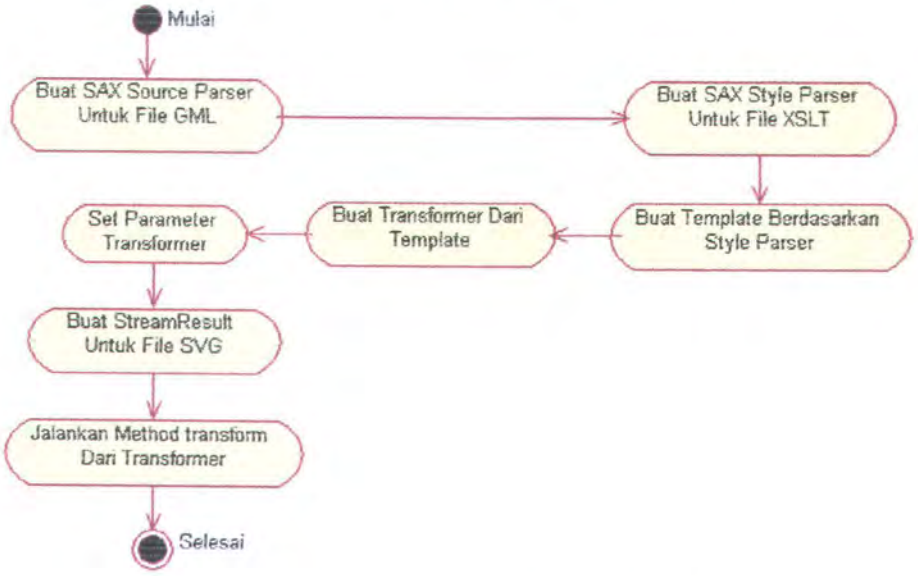

**Gambar 3. 5 Diagram Aktivitas Konversi *Shapefile* ke GML**

### 3.3.3.2 Aktivitas Konversi GML ke SVG

Pada proses ini dilakukan penerjemahan tag GML dengan tag XSLT menggunakan XSLT *processor* yang kemudian akan dikonversikan menjadi dokumen SVG. Berikut langkah-langkah yang dilalui pada rangkaian kegiatan di atas, seperti pada gambar 3.6 berikut.

Hal pertama yang dilakukan adalah inisialisasi SAXSource pertama yang berisi dokumen .gml yang ingin dikonversi. Inisialisasi juga SAXSource kedua yang berisi dokumen .xsl untuk pemrosesan dokumen .gml yang ingin dikonversi. Lalu buat Template yang berisi SAXSource XSLT yang telah dikompilasi. Selanjutnya inisialisasi Transformer yang akan mengopi isi Template (XSLT terkompilasi) ke dirinya. Kemudian set parameter dari isi Template tadi pada

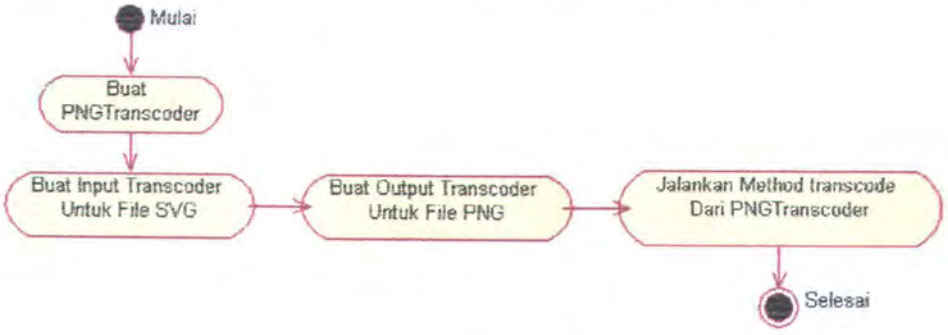
Transformer. Langkah selanjutnya adalah pembuatan *stream* untuk output *file*. Terakhir, jalankan proses transformasi milik Transformer.




**Gambar 3. 6 Diagram Aktivitas Konversi GML ke SVG**

### 3.3.3.3 Aktivitas Konversi SVG ke PNG

Pada proses pengonversian gambar dari tipe vektor SVG ke gambar dengan tipe *raster bitmap* (.png), yang pertama dilakukan adalah pembuatan PNGTranscoder. Kemudian inisialisasi input untuk transcoder yang berisi *path file* .svg. Inisialisasi juga output transcoder yang berisi *stream* untuk *file* .png. Setelah itu baru jalankan proses transcoding dari PNGTranscoder. Keseluruhan proses yang telah dijelaskan dapat dilihat pada gambar 3.7 di bawah.




**Gambar 3. 7 Diagram Aktivitas Konversi SVG ke PNG**

## 3.4 PERANCANGAN ANTAR-MUKA

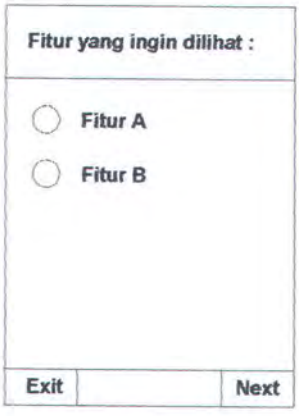
Antar-muka pada perangkat lunak ini hanya ada pada sisi *client* (ponsel) dan dibagi menjadi dua bagian, yaitu masukan dan keluaran.

Untuk menjalankan fungsi-fungsi yang terdapat pada aplikasi, pada tiap bagian antar-muka terdapat tombol-tombol yang dibuat sesederhana mungkin untuk menghindari kesalahan pengartian oleh pengguna, diantaranya adalah *Exit* untuk keluar dari aplikasi, *Next* untuk melanjutkan ke halaman selanjutnya, *Back* untuk kembali ke halaman sebelumnya, *View* untuk melihat peta, *Zoom* akan mengaktifkan kursor area untuk memilih area pada peta yang ingin dilihat lebih dekat, dan *Cancel* untuk membatalkan sebuah perintah.

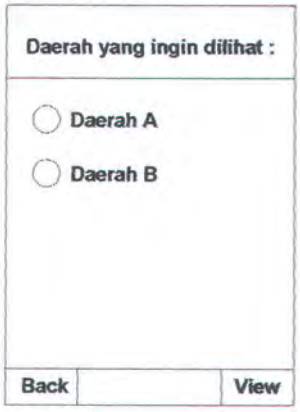
### 3.4.1 Antar-Muka Masukan

Pada antar-muka masukan, untuk meminimalkan kesalahan oleh pengguna yang mungkin terjadi maka bentuk *inputan* yang akan digunakan adalah *radiobutton* dan kursor area.

Bentuk *inputan radiobutton* digunakan pada waktu pemilihan fitur dan daerah yang ingin ditampilkan pada peta oleh pengguna seperti yang dapat dilihat pada gambar 3.8 dan gambar 3.9.




**Gambar 3. 8 Rancangan Antar-Muka Pemilihan Fitur**




**Gambar 3. 9 Rancangan Antar-Muka Pemilihan Daerah**

Sedangkan bentuk *inputan* kursor area digunakan pada waktu pemilihan area pada peta yang ingin dilihat lebih dekat (*zoom*), seperti yang terlihat pada gambar 3.10.


**Gambar 3. 10 Rancangan Antar-Muka Pemilihan Area *Zoom***

Dengan menggunakan kedua bentuk *inputan* di atas maka pengguna hanya bisa memilih dari yang telah tersedia dan dengan begitu maka aplikasi akan terhindar dari kemungkinan kesalahan masukan yang diberikan pengguna.

### 3.4.2 Antar-Muka Keluaran

Pada antar-muka keluaran, seperti yang dapat dilihat pada gambar 3.11, pengguna dapat melihat peta daerah dan fitur yang telah dipilihnya dengan ukuran yang telah disesuaikan menurut lebar ponsel yang digunakannya.


**Gambar 3. 11 Rancangan Antar-Muka Keluaran**

# BAB IV

# IMPLEMENTASI PERANGKAT LUNAK

# BAB IV
# IMPLEMENTASI PERANGKAT LUNAK

Berdasarkan rancangan perangkat lunak yang telah dibuat pada bab sebelumnya, maka tahapan selanjutnya adalah implementasi perangkat lunak. Pada tahapan ini akan diberikan realisasi dari rancangan keseluruhan proses yang terjadi pada aplikasi, baik di sisi *server* maupun di sisi *client*, serta bentuk implementasi antar-muka aplikasi di sisi *client*.

## 3.5 LINGKUNGAN IMPLEMENTASI

Perangkat lunak ini dibagi menjadi dua bagian, yaitu *server* dan *client*. Kedua bagian aplikasi dikembangkan dalam *platform* Java. Pada *server*, implementasi dilakukan dengan menggunakan JDK 1.5.0 serta tambahan *library* dari Deegree, Saxon, dan Batik. Untuk pemrograman web digunakan JSP (*JavaServerPage*) dengan web *server* jakarta-tomcat-4.1.20. Sedangkan pada *client*, implementasi dilakukan dengan menggunakan simulator dari J2ME Wireless Toolkit 2.2 dan *library* MIDP 2.0.

Perangkat lunak yang digunakan sebagai editor pada tahap implementasi ini adalah JCreator 2.00 Pro, Macromedia Dreamweaver MX, dan ArcView 3.1 serta emulator dari J2ME WTK 2.2.

Sedangkan perangkat keras yang digunakan dalam penyelesaian tahap ini adalah komputer dengan spesifikasi sebagai berikut:

- *Processor* : AMD Duron 1.80 GHz.
- *Memory* : 256 MB.
- HDD : 30 GB.

## 3.6 IMPLEMENTASI PROSES

Implementasi proses pada aplikasi yang akan dijelaskan di sini adalah realisasi dari keseluruhan proses yang terdapat pada perangkat lunak.

### 3.6.1 Proses Lihat Peta

Fungsionalitas ini diawali dari pengguna yang memilih fitur dan daerah yang ingin ditampilkan pada peta. Setelah data fitur dan daerah telah dimasukkan oleh pengguna dengan menggunakan *radiobutton*, `Mapviewer` kemudian akan mengolah data tersebut menjadi *query* untuk url. Cara melakukan pengolahan dapat dilihat pada gambar 4.1.

```
private String setQuery()
{
      String query = "";

      query += "fitur=" + radioButtonFitur.getSelected();
      query += "&daerah=" + radioButtonDaerah.getSelected();

      return query;
}
```

**Gambar 4. 1 *Method* Untuk Mengolah *Query* Lihat Peta**

Setelah itu *query* dari `MapViewer` tadi akan ditambahkan ke url pada `Connection2MapServer` dengan menggunakan `setURL(String url_query)`. Baru setelah itu, seperti yang dapat dilihat pada gambar 4.2, koneksi ke server dapat dibuat ke url yang telah di-set.

```
private void createConnection()
{
      Try
      {
            conn = (HttpConnection)Connector.open( url );
            ...
      }
      catch(ConnectionNotFoundException cnfe)
      {
            notify.setString( "Koneksi ke Server Terputus" );
            phoneDisplay.setCurrent( notify, selectFitur );
      }
}
```

**Gambar 4. 2 *Method* Untuk Membuat Koneksi**

Data dari pengguna dalam bentuk parameter *query* url yang telah dikirimkan tadi kemudian diambil oleh *class* `view` untuk diolah menjadi bentuk `String` *path* dari *shapefile-shapefile* yang diperlukan. Setelah itu semua *shapefile* tadi akan dikonversi menjadi *file* .gml dan oleh *class* `GML2SVG`, *file* .gml

yang telah didapat dikonversi lagi menjadi *file* .svg dengan bantuan XSLT. *File* .svg yang dihasilkan terakhir akan dikonversi menjadi *file* .png oleh *class* SVG2PNG. Langkah-langkah mulai dari pengambilan parameter sampai proses konversi selesai dapat dilihat pada gambar 4.3. Untuk mengetahui lebih detil tentang proses konversi sendiri akan dijelaskan pada sub bab tersendiri di bawah.

```
...
String fitur = getParameter( "fitur" );
String area = getParameter( "area" );

String fileSHP1 = area + ".shp";
String fileSHP2 = fitur + ".shp";
String fileSHP3 = jalan + ".shp";

shp2gml.addFiles2Vector( fileSHP1 );
shp2gml. addFiles2Vector( fileSHP2 );
shp2gml. addFiles2Vector( fileSHP3 );

shp2gml.convert( getFiles2Convert(), fileGML );
gml2svg.convert( fileGML, fileXSL, fileSVG, lebarHP );
svg2png.convert( fileSVG, filePNG );
...
```

**Gambar 4. 3 Pengambilan Parameter Lihat Peta dan Pemanggilan *Method* Konversi Melalui *Class view***

*File* .png inilah yang akan dibaca dan diubah dalam bentuk `Byte` untuk kemudian dijadikan `Image` oleh *class* `Connection2MapServer`. Selanjutnya `Image` ini diletakkan dalam `Canvas` oleh `MapViewer`. `Canvas` dengan `Image` dari *file* .png inilah yang akan dilihat oleh pengguna pada ponselnya. Pemrosesan *file* .png untuk ditampilkan ke layar ponsel ini dapat dilihat pada gambar 4.4.

```
...
inputStream = connection.openDataInputStream();
inputStream.readFully( byteArray );

image = createImage( byteArray );
mapViewer.setMap( image );

canvas = new MyCanvas( map );
phoneDisplay.setCurrent( canvas );
...
```

**Gambar 4. 4 Pemrosesan *File* PNG Untuk Dapat Ditampilkan ke Ponsel**

### 3.6.2 Proses *Zoom*

Proses ini diawali dari pengguna yang memilih area pada peta yang ingin dilihat secara lebih dekat. Setelah itu data koordinat area *zoom* yang diinginkan dimasukkan oleh pengguna dengan menggunakan kursor area dan diolah menjadi *query* seperti yang terlihat pada gambar 4.5 di bawah.

```
private String setQuery()
{
      String query = "";

      query += "x1=" + x1Kursor + "&y1=" + y1Kursor;
      query += "x2=" + x2Kursor + "&y2=" + y2Kursor;

      return query;
}
```

**Gambar 4. 5 *Method* Untuk Mengolah *Query Zoom***

*Query* di atas lalu ditambahkan ke url untuk koneksi ke server. *Method* yang digunakan untuk koneksi ini telah dijelaskan pada proses Lihat Peta.

Selanjutnya, data dalam bentuk parameter *query* url akan diambil oleh *class* `zoom` untuk diolah menjadi variabel-variabel yang diperlukan dalam proses ini. Variabel-variabel yang didapatkan kemudian digunakan sebagai parameter pada *method* `zoom()` `GML2SVG` untuk mengubah *file* .gml sesuai koordinat *zoom* dari pengguna. *File* .gml yang telah diubah kemudian dikonversi menjadi *file* .svg dengan bantuan XSLT. Setelah itu *file* .svg yang dihasilkan dikonversi menjadi *file* .png oleh `SVG2PNG`. Proses pengambilan parameter, pemanggilan *method zoom*, dan konversi ini dapat dilihat pada gambar 4.6 di bawah.

```
...
Int x1 = Integer.parseInt( getParameter("x1") );
int y1 = Integer.parseInt( getParameter("y1") );
int x2 = Integer.parseInt( getParameter("x2") );
int y2 = Integer.parseInt( getParameter("y2") );

gml2svg.zoom( fileGML, img_w, img_h, x1, y1, x2, y2);
gml2svg.convert( fileGML, fileXSL, fileSVG, lebarHP );
svg2png.convert( fileSVG, filePNG );
...
```

**Gambar 4. 6 Pengambilan Parameter *Zoom* dan Pemanggilan *Method* Konversi Melalui *Class zoom***

Karena `Canvas` yang akan ditampilkan telah dibuat dari proses Lihat Peta sebelumnya, maka yang perlu dilakukan pada langkah terakhir ini hanyalah mengeset `Image` pada `Canvas`, seperti yang dapat dilihat pada gambar 4.7.

```
...
canvas.setImage( zoomed_map );
phoneDisplay.setCurrent( canvas );
...
```

**Gambar 4. 7 Pengesetan *Image* Pada Canvas**

### 3.6.3 Proses Konversi *Shapefile* ke GML

Langkah pertama dalam proses konversi *shapefile* ke format GML ini adalah untuk mendapatkan titik-titik koordinat yang membatasi obyek-obyek geometri pada *shapefile*, seperti yang diperlihatkan *method* di gambar 4.8.

```
private void setBox( String SHPFilePath ) throws Exception
{
      ShapeFile sf = new ShapeFile( SHPFilePath );

      minX = sf.getFileMBR().getMin().getX();
      minY = sf.getFileMBR().getMin().getY();
      maxX = sf.getFileMBR().getMax().getX();
      maxY = sf.getFileMBR().getMax().getY();

      minX = Math.floor( minX ); minY = Math.floor( minY );
      maxX = Math.ceil( maxX );  maxY = Math.ceil( maxY );
}
```

**Gambar 4. 8 *Method* Untuk Mendapatkan Koordinat Pembatas *Shapefile***

Kemudian set sistem koordinat berdasarkan CRS (*Coordinate Reference System*) seperti dapat dilihat pada gambar 4.9.

```
...

private static final String CRS = "EPSG:4326";

ConvenienceCSFactory fac =
ConvenienceCSFactory.getInstance();
CoordinateSystem cs_ = fac.getCSByName( CRS );
Adapters ada = Adapters.getDefault();
CS_CoordinateSystem cs = ada.export( cs_ );

...
```

**Gambar 4. 9 Pengesetan Sistem Koordinat Berdasarkan CRS**

Langkah selanjutnya adalah mengambil fitur-fitur yang terdapat dalam *shapefile* dan menyimpan fitur-fitur tersebut dalam `Object`. Untuk melakukan hal ini, seperti yang dapat dilihat pada gambar 4.10, diperlukan inisialisasi *class* `Shapefile` dan *method* dari *class-class* `Feature`, `FeatureType`, dan `FeatureTypeProperty`. *Class-class* tersebut akan mengambil fitur-fitur dari `Shapefile` dan menyimpannya dalam bentuk *array*. Setelah itu tinggal mengambil fitur pada *array* tersebut dan menyimpannya dalam `Object`.

```
...
ShapeFile sf = new ShapeFile( String SHPFilePath );

for ( int i=1; i<=sf.getRecordNum(); i++ )
{
      Feature feature = sf.getFeatureByRecNo(i);
      FeatureType ftype = feature.getFeatureType();
      FeatureTypeProperty[] ftp = ftype.getProperties();

      for ( int j=0; j<ftp.length; j++ )
      {
          Object o = feature.getProperty( j );

          ...
      }
}
```

**Gambar 4. 10 Pengambilan dan Penyimpanan Fitur *Shapefile* Dalam Object**

Alasan mengapa fitur-fitur yang diambil dari *shapefile* harus disimpan dalam `Object` adalah karena kemudian fitur-fitur tersebut akan dikelompokkan berdasarkan properti geometri dan non-geometrinya. Pengelompokan tersebut, seperti dapat dilihat pada gambar 4.11 dan gambar 4.12, dilakukan menggunakan pembandingan dengan *class* `GM_Object`.

```
...

if ( o instanceof GM_Object )
{
    ((GM_Object_Impl)o).setCoordinateSystem( cs );
    String s = GMLAdapter.export( (GM_Object)o );

    ...
}
```

**Gambar 4. 11 Pengelompokan Object Berdasarkan Properti Geometri**

Sedangkan untuk penyimpanannya dalam `String` menggunakan *class* `GMLAdapter` jika termasuk properti geometri dan *class* `XMLTools` untuk properti non-geometri.

```
...

else
{
      String s =
      XMLTools.validateCDATA( o.toString() ).toString();
}
...
```

**Gambar 4. 12 Pengelompokan Object Berdasarkan Properti Non-Geometri**

Langkah terakhir pada proses konversi ini adalah pengubahan format dari dokumen teks biasa menjadi format dokumen XML, seperti yang ditunjukkan gambar 4.13. Pada urutan langkah di bawah dapat dilihat bahwa pembuatan dokumen *inputan* berisi bentuk `String` dari `gml`. `Gml` adalah *class* `StringBuffer` yang telah diinisialisasi pada awal proses konversi yang berfungsi sebagai tempat penggabungan `String-String` yang digunakan untuk menyimpan tag-tag GML yang belum terformat, selama proses konversi, seperti koordinat pembatas *shapefile*, CRS, serta terakhir fitur-fitur geometri dan non-geometri *shapefile*.

```
...

Document doc =
XMLTools.parse( new StringReader( gml.toString() ) );
Source xmlSource = new DOMSource( doc );
Result result =
new StreamResult( new FileOutputStream( GMLFilePath ) );

TransformerFactory transformerFactory =
TransformerFactory.newInstance();
Transformer transformer =
transformerFactory.newTransformer();

transformer.setOutputProperty( "indent", "yes" );
transformer.transform( xmlSource, result );

...
```

**Gambar 4. 13 Pengubahan Format Dokumen Menjadi XML**

*Class-class* yang digunakan dalam proses pengambilan fitur-fitur *shapefile* di atas, seperti `Shapefile`, `FeatureTypeProperty`, dan `Object_GM`, adalah bagian dari *library* Deegree.

### 3.6.4 Proses Konversi GML ke SVG

Hal pertama yang dilakukan pada proses konversi GML ke SVG ini, seperti dapat dilihat pada gambar 4.14, adalah inisialisasi *class* `SAXSource` pertama yang berisi dokumen GML yang ingin dikonversi. Dokumen GML tadi sebelumnya harus dimasukkan terlebih dahulu ke dalam *class* `ExtendedInputSource`. Pada inisialisasi `SAXSource` di atas juga dicantumkan penggunaannya sebagai elemen *source* proses konversi ini dengan menggunakan *method* `getSourceParser()` dari *class* `TransformerFactoryImpl`.

```
...
File gml = new File( GMLFilePath );
TransformerFactoryImpl fact = new TransformerFactoryImpl();

ExtendedInputSource eis = new ExtendedInputSource( gml );
SAXSource src = new SAXSource( fact.getSourceParser(),eis );
extendedinputsource.setEstimatedLength( (int)gml.length() );
...
```

**Gambar 4. 14 Pembuatan SAX Source Parser Untuk File GML**

Kemudian inisialisasi juga `SAXSource` kedua, seperti terlihat pada gambar 4.15, yang berisi dokumen XSLT untuk pemrosesan dokumen GML yang ingin dikonversi. Dokumen XSLT tadi sebelumnya juga harus dimasukkan terlebih dahulu ke dalam *class* `ExtendedInputSource`. Pada inisialisasi `SAXSource` kedua ini juga dicantumkan penggunaannya sebagai elemen *style* proses konversi ini dengan menggunakan *method* `getStyleParser()` yang didapatkan dari *class* `TransformerFactoryImpl`.

```
...
File xslt = new File( XSLTFilePath );

ExtendedInputSource eis2 = new ExtendedInputSource( xslt );
SAXSource sty = new SAXSource( fact.getStyleParser(),eis2 );
...
```

**Gambar 4. 15 Pembuatan SAX Style Parser Untuk File XSLT**

Langkah selanjutnya, seperti yang dapat dilihat pada gambar 4.16, adalah inisialisasi *class* `Templates` yang berisi SAXSource XSLT yang telah dikompilasi. Lalu, dengan memanfaatkan *class* `Templates`, inisialisasi juga *Class* `Transformer` yang akan mengopi isi `Templates` (xslt terkompilasi) ke dirinya. Setelah itu set parameter dari isi `Templates` tadi pada `Transformer` dengan menggunakan *class* `Controller`. Langkah selanjutnya adalah pembuatan *stream* untuk output *file* .svg yang telah diinisalisasi sebelumnya dengan *class* `StreamResult`. Terakhir, jalankan *method* `transform` dari `Transformer`.

```
...
File svg = new File( SVGFilePath );
ParameterSet param = new ParameterSet();
Templates temp = fact.newTemplates( sty );

Transformer trans = temp.newTransformer();
((Controller) trans).setParams( param );
StreamResult output = new StreamResult( svg );
Trans.transform( src, output );
...
```

**Gambar 4. 16 Proses Transformasi GML ke SVG Menggunakan XSLT**

*Class-class* yang digunakan dalam proses transformasi GML di atas, seperti `Transformer`, `ParameterSet`, dan `SAXSource`, adalah bagian dari *library* Saxon.

### 3.6.5 Proses Konversi SVG ke PNG

Langkah-langkah yang dilakukan pada proses konversi vektor ke *bitmap* ini, seperti yang dapat dilihat pada gambar 4.17 adalah, pertama pembuatan *instance* dari *class* `PNGTranscoder`. Setelah itu inisialisasi *input* untuk *transcoder* menggunakan *class* `TranscoderInput` dengan parameter yang berisi *path file* .svg. Inisialisasi juga, dengan menggunakan `TranscoderOutput`, *output* untuk *transcoder* dengan parameter yang berisi *stream* untuk *file* .png. Kemudian *method* `transcode` dari *class* `PNGTranscoder` dapat dijalankan dengan parameter *input* dan *output* yang telah dibuat.

```
...
PNGTranscoder trans = new PNGTranscoder();

TranscoderInput input = new TranscoderInput( SVGFilePath );

OutputStream ostream = new FileOutputStream( PNGFilePath );
TranscoderOutput output = new TranscoderOutput( ostream );

trans.transcode(input, output);
...
```

**Gambar 4. 17 Proses *Transcoding* SVG ke PNG**

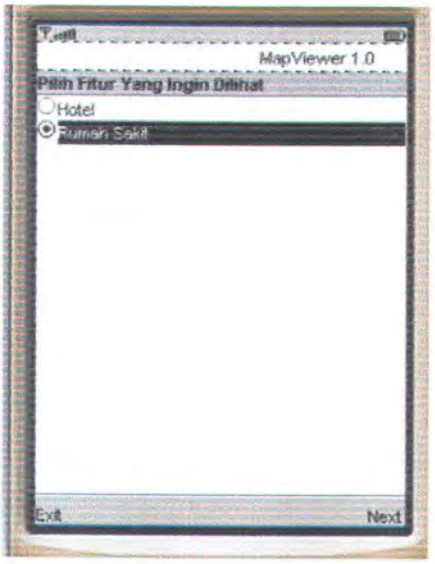
*Class-class* yang digunakan dalam proses *transcoding* SVG di atas, seperti `PNGTranscoder`, `TranscoderInput`, dan `TranscoderOutput`, adalah bagian dari *library* Batik.

## 3.7 IMPLEMENTASI ANTAR-MUKA

Sebagaimana yang telah disebutkan pada awal bab, bagian ini akan memberikan gambaran tentang implementasi antar-muka aplikasi di sisi *client*.

### 3.7.1 Antar-Muka Masukan

Gambar 4.18 di bawah adalah apa yang akan dilihat pertama kali oleh pengguna ketika menjalankan perangkat lunak ini.




**Gambar 4. 18 Antar-Muka Pemilihan Fitur**

Antar-muka tersebut berfungsi sebagai bagian yang mana pengguna dapat memasukkan fitur pilihan yang ingin dilihatnya. Seperti yang telah disebutkan pada Bab Perancangan Perangkat Lunak, *inputan* yang ada hanya tersedia dalam bentuk *radiobutton*. Dengan begitu pengguna diharapkan tidak akan dibingungkan dalam pemakaiannya dan sekaligus menghindarkan aplikasi dari kemungkinan kesalahan *inputan* oleh pengguna. Tombol *Exit* pada bagian bawah perangkat lunak berfungsi jika pengguna ingin keluar dari aplikasi.

Setelah pengguna memilih fitur yang ingin dilihatnya dan menekan tombol *Next*, halaman yang akan muncul berikutnya adalah seperti yang dapat dilihat pada gambar 4.19 di bawah.

Antar-muka selanjutnya ini adalah bagian yang mana pengguna dapat memillih daerah yang ingin dilihatnya. Sebagaimana yang terdapat pada antar-muka sebelumnya, bagian ini pun juga hanya memiliki *radiobutton* sebagai satu-satunya bentuk *inputan* yang tersedia. Pada bagian ini pengguna dapat kembali ke halaman sebelumnya, Pemilihan Fitur, dengan menekan tombol *Back*.


**Gambar 4. 19 Antar-Muka Pemilihan Daerah**

Bentuk *inputan* selanjutnya, kursor area, seperti yang dapat dilihat pada gambar 4.20, dapat digerakkan jika tombol *Zoom* pada bagian bawah antar-muka keluaran telah ditekan.


**Gambar 4. 20 Antar-Muka Pemilihan Area *Zoom***

Bentuk *inputan* ini memungkinkan pengguna untuk memilih area pada peta yang ingin dilihatnya lebih dekat (*zoom*), sesuai dengan cakupan bidang di dalam bingkai segi empat (kursor area) yang dapat digerakkannya. Bila area pada peta yang ingin dilihat lebih dekat telah dipilih, maka pengguna dapat menekan tombol *View* yang terletak di bagian bawah. Sedangkan tombol *Cancel* berfungsi jika pengguna membatalkan keinginannya untuk menggunakan fungsi *zoom*.

### 3.7.2 Antar-Muka Keluaran

Gambar 4.21 di bawah adalah bentuk antar-muka keluaran yang muncul setelah pengguna memilih fitur serta daerah yang ingin dilihatnya dan menekan tombol *View*. Hasilnya, seperti yang dapat dilihat adalah peta dari sebuah daerah beserta fitur, seperti rumah sakit atau hotel, yang terdapat di daerah tersebut, sesuai dengan apa yang telah dipilih oleh pengguna pada proses sebelumnya. Antar-muka keluaran juga muncul setelah proses *zoom* yang mana pengguna akan memilih area dari peta yang ingin dilihatnya lebih dekat dengan menggunakan kursor area. Kursor tersebut, seperti telah dikatakan pada sub bab sebelum ini, baru akan dapat digerakkan setelah pengguna menekan tombol *Zoom*.


**Gambar 4. 21 Antar-Muka Keluaran**

# BAB V

# UJICOBA DAN EVALUASI

# BAB V
# UJI COBA DAN EVALUASI

Pada bagian ini akan dijelaskan tentang pelaksanaan uji coba terhadap proses yang ada pada perangkat lunak beserta evaluasinya. Uji coba ini dilakukan berdasarkan permasalahan yang diangkat dalam tugas akhir dan jawabannya. Dengan begitu diharapkan simpulan yang dihasilkan pada akhir pengerjaan tugas akhir ini merupakan hasil yang cukup valid dan telah melalui serangkaian proses pengujian dan evaluasi.

## 5.1 LINGKUNGAN UJI COBA

Pelaksanaan tahap uji coba terhadap proses aplikasi akan dilakukan pada sistem dengan spesifikasi sebagai berikut:

- *Processor* : AMD Duron 1.80 GHz.
- *Memory* : 256 MB.
- HDD : 30 GB.

Sedangkan perangkat lunak yang digunakan untuk menampilkan dan melakukan pembandingan pada tahap ini adalah sebagai berikut:

| | |
|---|---|
| *Emulator* | : J2ME WTK 2.2 |
| *Web* Browser | : Internet Explorer 6.0, Squiggle Batik Browser |
| *Image Comparer* | : ImageDiff |
| *Shapefile Viewer* | : ArcView 3.1 |
| *Image Manipulator* | : Microsoft Paint 5.1 |
| *Image Capturer* | : ACDSee 4.0 |

## 5.2 SKENARIO UJI COBA

Berdasarkan permasalahan yang menjadi fokus pengerjaan tugas akhir, maka dibuatlah dua skenario uji coba untuk melakukan proses pengujian terhadap jawaban yang telah dihasilkan dan sekaligus terhadap proses yang ada pada perangkat lunak. Jawaban tersebut didapatkan setelah melalui tahapan-tahapan dalam pengerjaan tugas akhir ini.

### 3.7.3 Skenario Uji Coba Konversi Format Data Spasial ke XML

Skenario ini dirancang untuk melakukan pengujian terhadap validitas proses konversi dari format data spasial menjadi format data XML yang terdapat pada perangkat lunak. Hasil yang diharapkan dari skenario ini ialah, apakah proses konversi telah berhasil dilakukan. Parameter yang harus dipenuhi sehingga proses konversi di sini dapat dikatakan berhasil adalah, jika proses tersebut telah mengindikasikan bahwa data yang ada pada format data spasial sebagai *input*, dan data yang terdapat pada format data XML sebagai *output*, benar-benar sama.

Proses pengujian yang dilakukan pada skenario ini untuk mendapatkan hasil yang diinginkan adalah, melalui pembandingan antara data atribut atau fitur non-geometri yang ada dalam format *shapefile* ESRI sebagai *input*, dan data dari *file* .gml sebagai *output* dari proses konversi. Jika semua data atribut yang ada dalam format *shapefile* ESRI didapati pula dalam *file* .gml dan jumlah *record* yang terdapat pada kedua *file* juga sama, maka proses dapat dikatakan berhasil.

### 3.7.4 Skenario Uji Coba Konversi Format XML ke *Bitmap*

Skenario ini dirancang untuk melakukan pengujian terhadap validitas proses konversi dari format XML menjadi format *bitmap* yang terdapat pada perangkat lunak. Hasil yang diharapkan dari skenario ini ialah, apakah proses konversi telah berhasil dilakukan. Parameter yang harus dipenuhi sehingga proses konversi di sini dapat dikatakan berhasil adalah, jika proses tersebut telah mengindikasikan bahwa gambar bertipe vektor dengan format XML sebagai *input*, dan gambar bertipe *raster* dengan format *bitmap* sebagai *output*, merupakan gambar yang identik.

Proses pengujian yang dilakukan pada skenario ini untuk mendapatkan hasil yang diinginkan adalah, melalui pembandingan antara gambar peta pada *file* .svg sebagai *input*, dan gambar peta pada *file* .png sebagai *output* dari proses konversi. Jika gambar peta pada *file* .png tidak mengalami banyak perubahan dibandingkan dengan gambar peta pada *file* .svg, maka proses dapat dikatakan berhasil.

## 5.3 PROSES UJI COBA

Sesuai dengan skenario uji coba yang telah didefinisikan pada sub bab di atas, maka dilakukan proses pengujian yang hasilnya nanti diharapkan akan dapat menjadi bahan evaluasi bagi tugas akhir ini.

### 5.3.1 Uji Coba Konversi Format Data Spasial ke XML

Sampel data yang digunakan pada uji coba kali ini adalah hasil *capture* atribut atau fitur non-geometri dari *file* Rs_mangu.shp (Gambar 5.1) dan atribut dari *file* Pasar_taman.shp (Gambar 5.3) pada ArcView 3.1, serta hasil *capture* sebagian dari *file* Rs_mangu.gml (Gambar 5.2) dan *file* Pasar_taman.gml (Gambar 5.4).

Dapat dilihat bahwa pada gambar 5.1 dan gambar 5.2, seluruh atribut atau fitur non-geometri dari Rs-mangu.shp, seperti Tipe, Nama, Desa_kel, Kecamatan, Bangunan, Halaman, dan T_tidur, juga terdapat pada *file* Rs_mangu.gml.

Attributes of Rs_mangu.shp

| Shape | Tipe | Nama | Desa_kel | Kecamatan | Bangunan | Halaman | T_tidur |
|---|---|---|---|---|---|---|---|
| Point | Rumah sakit | RS Islamiyah | Madiun Lor | Manguharjo | 675 | 350 | 75 |
| Point | Rumah sakit | RS Santa Clara | Pangongangan | Manguharjo | 520 | 240 | 78 |

**Gambar 5. 1 Atribut Rs_mangu.shp Dengan *Viewer* ArcView 3.1**

```
- <gml:featureMember>
  - <exp:Titik ID="2:0">
      <tipe>Rumah sakit</tipe>
      <nama>RS Islamiyah</nama>
      <desa_kel>Madiun Lor</desa_kel>
      <kecamatan>Manguharjo</kecamatan>
      <bangunan>675</bangunan>
      <halaman>350</halaman>
      <t_tidur>75</t_tidur>
    + <gml:location>
    </exp:Titik>
  </gml:featureMember>
- <gml:featureMember>
  - <exp:Titik ID="2:1">
      <tipe>Rumah sakit</tipe>
      <nama>RS Santa Clara</nama>
      <desa_kel>Pangongangan</desa_kel>
      <kecamatan>Manguharjo</kecamatan>
      <bangunan>520</bangunan>
      <halaman>240</halaman>
      <t_tidur>78</t_tidur>
    + <gml:location>
    </exp:Titik>
  </gml:featureMember>
```

**Gambar 5. 2 Format XML Atribut Rs_mangu.shp**

Dapat dilihat juga bahwa pada gambar 5.3 dan gambar 5.4, seluruh atribut atau fitur non-geometri dari Pasar-taman.shp, seperti Nama, Desa_kel, Kecamatan, Bangunan, dan Halaman, terdapat pula pada *file* Pasar_taman.gml. Selain itu, baik pada contoh pertama maupun pada contoh kedua ini jumlah *record* yang dimiliki kedua *file* (.shp dan .gml) juga sama.

Attributes of Pasar_taman.shp

| Shape | Nama | Desa_kel | Kecamatan | Bangunan | Halaman |
|---|---|---|---|---|---|
| Point | Pasar Sleko | Pandean | Taman | 6143 | 522 |
| Point | Pasar Besar | Taman | Taman | 6611 | 662 |
| Point | Pasar Lama | Kejuron | Taman | 1568 | 161 |

**Gambar 5. 3 Atribut Pasar_taman.shp Dengan *Viewer* ArcView 3.1**

```
- <gml:featureMember>
  - <exp:Titik ID="2:0">
      <nama>Pasar Sleko</nama>
      <desa_kel>Pandean</desa_kel>
      <kecamatan>Taman</kecamatan>
      <bangunan>6143</bangunan>
      <halaman>522</halaman>
    + <gml:location>
    </exp:Titik>
  </gml:featureMember>
- <gml:featureMember>
  - <exp:Titik ID="2:1">
      <nama>Pasar Besar</nama>
      <desa_kel>Taman</desa_kel>
      <kecamatan>Taman</kecamatan>
      <bangunan>6611</bangunan>
      <halaman>662</halaman>
    + <gml:location>
    </exp:Titik>
  </gml:featureMember>
- <gml:featureMember>
  - <exp:Titik ID="2:2">
      <nama>Pasar Lama</nama>
      <desa_kel>Kejuron</desa_kel>
      <kecamatan>Taman</kecamatan>
      <bangunan>1568</bangunan>
      <halaman>161</halaman>
    + <gml:location>
    </exp:Titik>
  </gml:featureMember>
```

**Gambar 5. 4 Format XML Atribut Pasar_taman.shp**

### 5.3.2 Uji Coba Konversi Format XML ke *Bitmap*

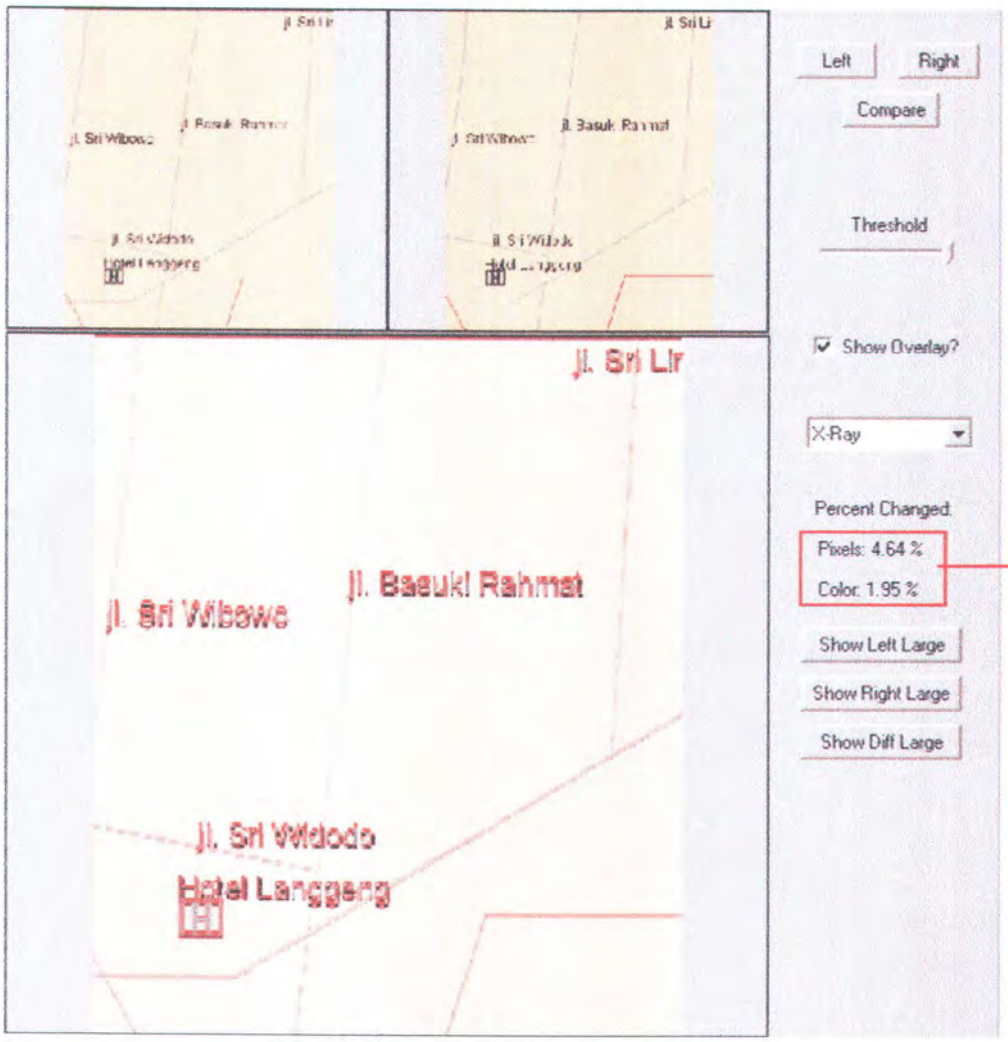
Sampel data yang digunakan pada uji coba kedua kali ini adalah dokumen hasil *capture* dari *file* .svg, serta dokumen *file* .png yang merupakan *output* dari proses konversi .svg ke .png. Keduanya dapat dilihat pada tabel 5.1 di bawah.

**Tabel 5. 1 *Thumbnail* Dokumen Uji Coba Konversi Format XML ke *Bitmap***

| Nama Dokumen | *Image* Vektor .svg (*Capture*) | *Image Bitmap* .png |
|---|---|---|
| inap_karto_zoomed |  |  |
| pasar_taman |  |  |
| pasar_taman_zoomed |  |  |
| rs_mangu |  |  |
| rs_mangu_zoomed |  |  |

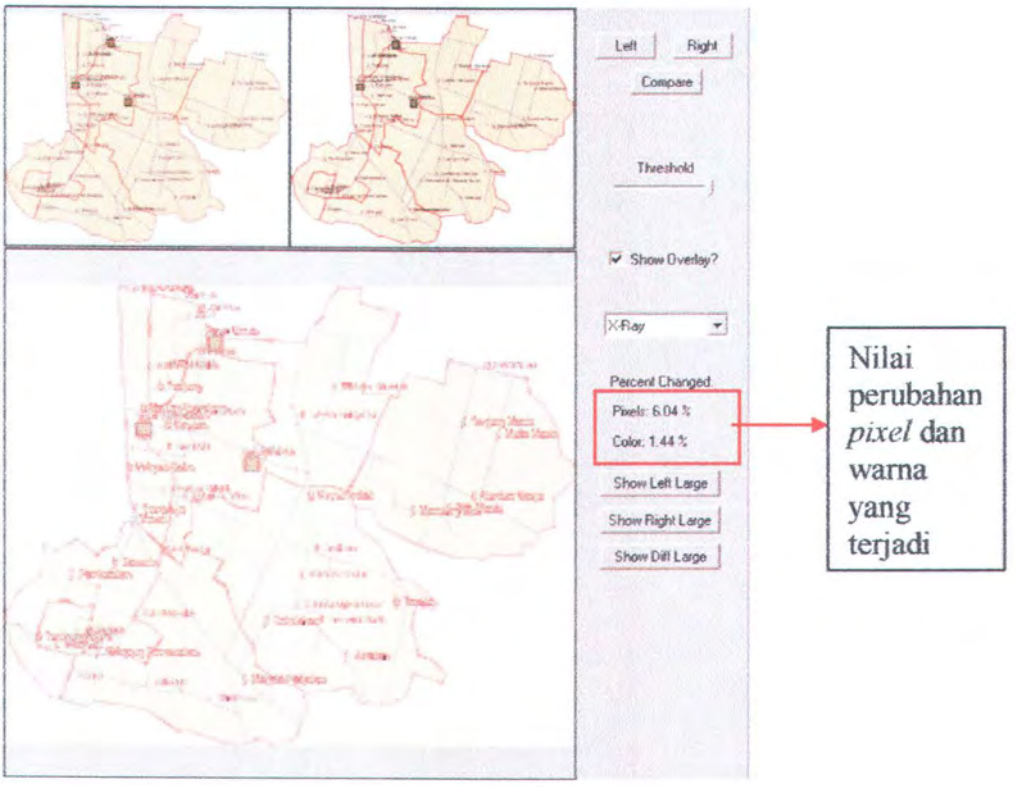
Pembandingan pasangan *file* pada tabel diawali dengan proses *capture* dokumen .svg pada *web browser* menggunakan ACDSee 4.0. Lalu, gambar hasil *capture* disamakan dimensi dan formatnya dengan gambar pada *file* .png, memanfaatkan perangkat lunak Microsoft Paint 5.1. Setelah kedua gambar memiliki format dan dimensi yang sama, barulah keduanya dibandingkan dengan menggunakan ImageDiff. Aplikasi yang digunakan dalam pembandingan gambar ini akan memerlihatkan tingkat (dalam persen) perubahan *pixel* dan warna, yang terjadi dari gambar satu ke gambar lainnya.

Berikut merupakan hasil *capture* dari aplikasi ImageDiff saat digunakan dalam proses pembandingan pasangan gambar pada tabel 5.1 di atas.

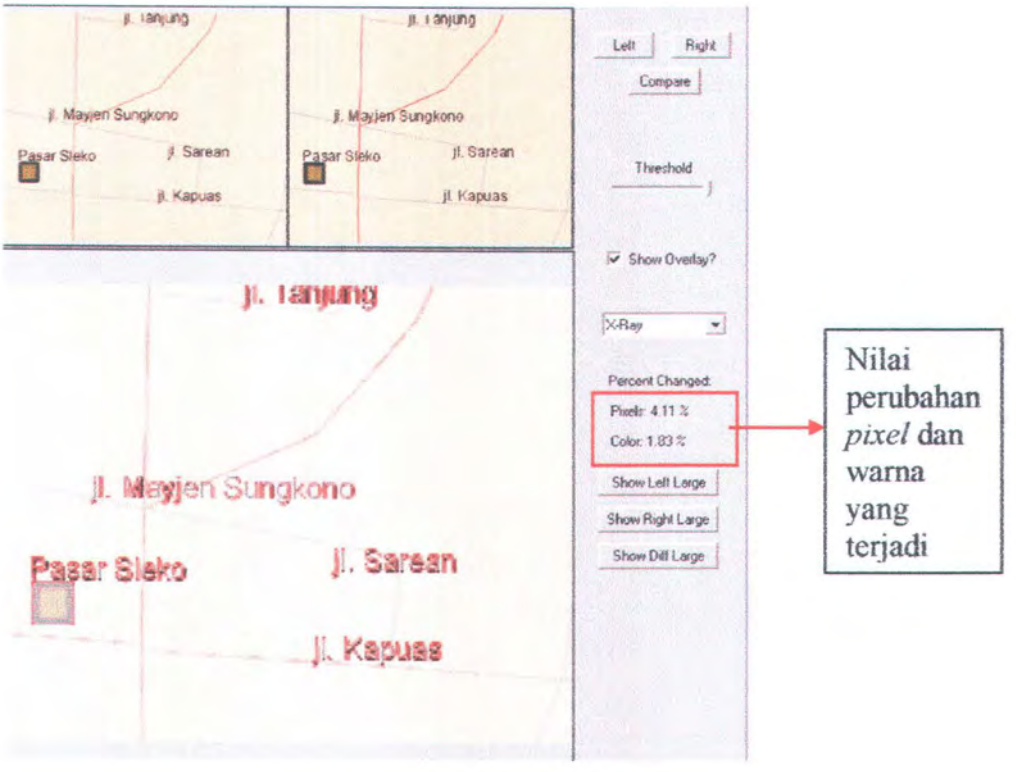



Nilai perubahan *pixel* dan warna yang terjadi

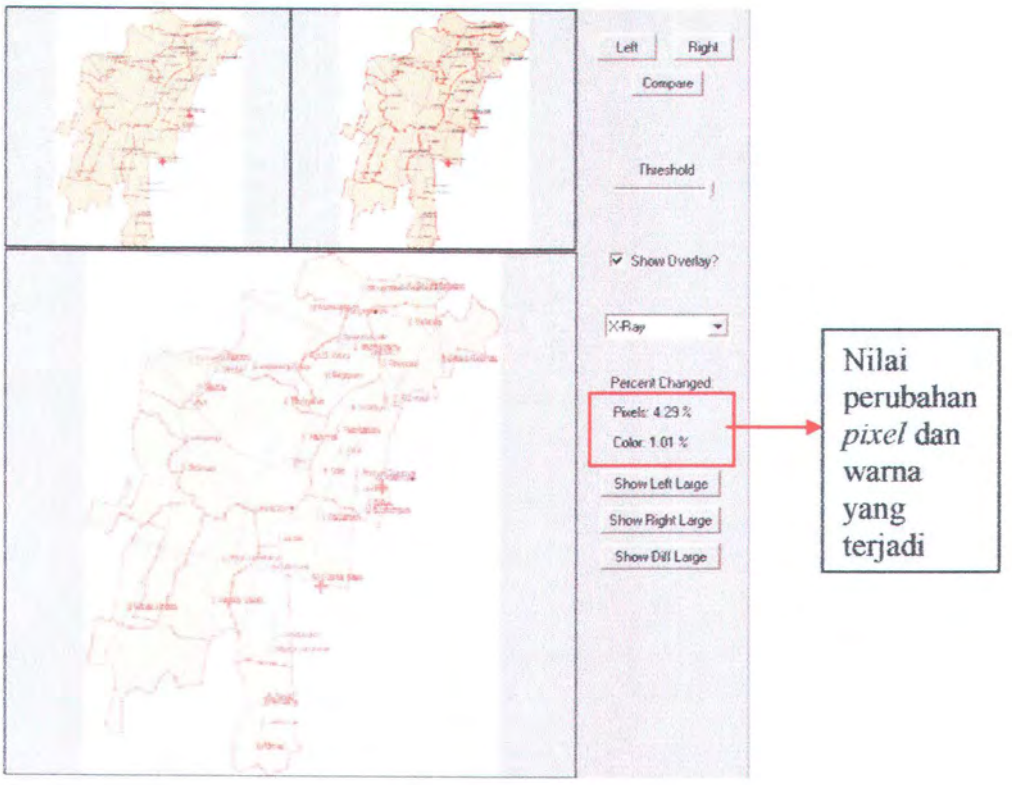
**Gambar 5. 5 Pembandingan *Image* Inap_karto_zoomed**



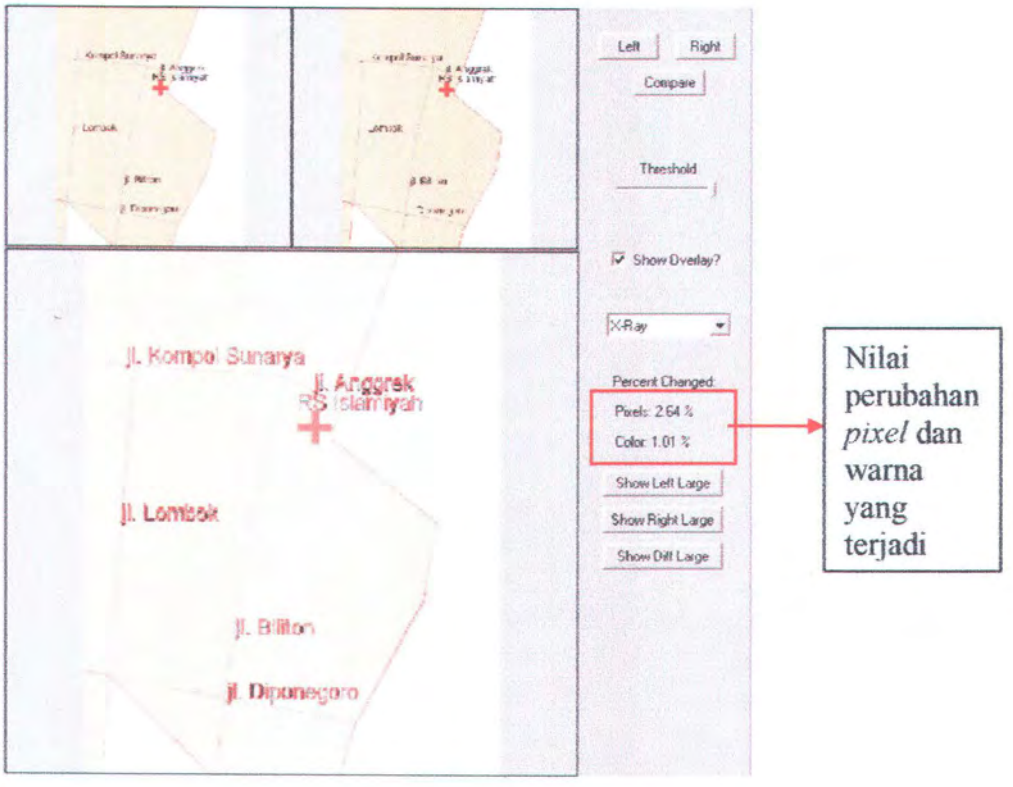

**Gambar 5. 6 Pembandingan *Image* Pasar_taman**




**Gambar 5. 7 Pembandingan *Image* Pasar_taman_zoomed**




**Gambar 5. 8 Pembandingan *Image* Rs_mangu**




**Gambar 5. 9 Pembandingan *Image* Rs_mangu_zoomed**

Hasil dari proses pembandingan sebelumnya yang telah dilakukan dengan menggunakan perangkat lunak ImageDiff dapat dilihat pada tabel 5.2 di bawah.

**Tabel 5. 2 Hasil Uji Coba Konversi Format XML ke *Bitmap***

| Dokumen *Image* | Perubahan (dalam %) | |
|---|---|---|
| | *Pixel* | Warna |
| inap_karto_zoomed | 4,64 | 1,95 |
| pasar_taman | 6.04 | 1.44 |
| pasar_taman_zoomed | 4.11 | 1.83 |
| rs_mangu | 4,29 | 1,01 |
| rs_mangu_zoomed | 2.64 | 1.01 |

## 5.4 EVALUASI

Setelah melalui serangkaian proses pengujian pada sub bab sebelumnya, selanjutnya akan dilakukan tahap evaluasi terhadap masing-masing hasil uji coba yang telah didapatkan.

### 5.4.1 Evaluasi Uji Coba Konversi Format Data Spasial ke XML

Dari pengujian konversi format data spasial ke XML, didapatkan hasil bahwa data atribut yang ada dalam format *shapefile* ESRI didapati pula dalam *file* .gml yang merupakan *file output* dari proses konversi tersebut. Selain itu jumlah *record* yang terdapat pada kedua *file* tersebut (.shp dan .gml) juga sama.

Dengan begitu dapat dilihat bahwa proses konversi format data spasial ke XML telah memenuhi parameter keberhasilan yang diberikan pada definisi skenario uji coba sebelumnya.

### 5.4.2 Evaluasi Uji Coba Konversi Format XML ke *Bitmap*

Dari pengujian konversi format XML ke *bitmap*, didapatkan hasil bahwa gambar peta pada *file* .png, yang merupakan *file output* dari proses konversi tersebut, tidak mengalami banyak perubahan dan identik dengan *capture* gambar peta pada *file* .svg.

Dengan begitu dapat dilihat bahwa proses konversi format XML ke *bitmap* telah memenuhi parameter keberhasilan yang diberikan pada definisi skenario uji coba sebelumnya.

# BAB VI

# PENUTUP

# BAB VI
# PENUTUP

Pada bab ini akan diberikan simpulan berupa jawaban dari permasalahan yang menjadi fokus utama pengerjaan tugas akhir beserta saran yang dapat dijadikan acuan pada pengembangan selanjutnya.

## 6.1 SIMPULAN

Berikut merupakan beberapa simpulan yang dapat diambil setelah menyelesaikan tahapan pengerjaan tugas akhir.

- Pengonversian format data spasial menjadi format data xml dapat dilakukan dengan menggunakan *library* dari Deegree dan Saxon. *Library* dari Deegree digunakan pada proses pengambilan fitur data spasial. Sedangkan *library* dari Saxon digunakan pada proses pengubahan format fitur yang diambil tersebut menjadi dokumen dengan format XML.
- Pengonversian format data XML menjadi format gambar *bitmap* dapat dilakukan dengan menggunakan XSLT serta *library* dari Saxon dan Batik. *Library* dari Saxon digunakan pada proses pengubahan format data XML menjadi format gambar vektor dengan bantuan XSLT. Sedangkan *library* dari Batik digunakan pada proses pengubahan format gambar vektor menjadi format gambar *bitmap*.
- Pemrosesan informasi dari *client* (ponsel) untuk diaplikasikan ke format data spasial di *server* dapat dilakukan dengan mengubah format data spasial tersebut menjadi format data XML terlebih dahulu. Mengingat pemrosesan format data spasial secara langsung sangat sulit untuk dapat dilakukan, maka data spasial tersebut dapat diubah terlebih dahulu ke format data XML menggunakan simpulan pertama. Selanjutnya barulah dilakukan pemrosesan informasi dari *client* ke data XML tadi.

## 6.2 SARAN

Perangkat lunak yang telah dikembangkan sampai saat ini tentu saja masih memiliki banyak kekurangan, untuk itu beberapa saran yang dapat diberikan untuk dijadikan acuan pada pengembangan selanjutnya antara lain adalah:

- Perbaikan untuk pengaturan tata letak dan penulisan pada nama jalan serta fitur peta yang ditampilkan, sehingga lebih mudah dibaca.
- Metode penghapusan *file* yang sudah tidak terpakai di *server* secara otomatis, sehingga tidak menghabiskan terlalu banyak ruang penyimpanan atau harus melakukannya secara manual.
- Riset mengenai metode konversi format data spasial ke format gambar vektor atau *raster* lainnya yang mungkin lebih cepat atau lebih efisien.

# DAFTAR PUSTAKA

# DAFTAR PUSTAKA


[1] _____, *Document Structure-SVG 1.1*, http://www.w3c.org/TR/ SVG11/, 2003

[2] _____, *ESRI Shapefile Technical Description*, ESRI White Paper, 1998

[3] _____, *GIS Concept Overview*, http://www.esri.com/software/arcgis/concepts/, ESRI

[4] Apache Software Foundation, *Batik Overview*. http://xml.apache.org/batik/, Apache, 2005

[5] Arif Wibisono, *Java Programming*, Laboratorium Pemrograman Sistem Informasi, 2005

[6] Beny Yulkurniawan Victorio Nasution, *Perancangan Dan Pembuatan Sistem Konversi Data Geografis Pada Rehabilitasi Hutan Dan Lahan Kabupaten Kutai Barat Berbasis Xml Dan Java*, Tugas Akhir, Sistem Informasi, Institut Teknologi Sepuluh Nopember, 2005

[7] Deegree Project Team, *Building Blocks for Spatial Data Infrastructures*, http://deegree.sourceforge.net/index.html, Deegree, 2003

[8] Gerald Bauer (Chairmain, CEO, CFO and CTO of Me, Myself & I, Inc.), *Scalable Vector Graphics (SVG) Creating High-End 2D Graphics Using XML*, Java User Group (JUG) Austria Talk, November 2002

[9] James Clark, *XSL Transformations (XSLT) Version 1.0*, http://www.w3.org/TR/1999/REC-xslt-19991116, W3C, 1999

[10] Kim Topley, *J2ME in a Nutshell*, O'Reilly, 2002

[11] Michael H Kay, *SAXON The XSLT and XQuery Processor*. http://saxon.sourceforge.net/, 2005

[12] Milan Trninic, *GML to SVG - how to?*, http://koala.ilog.fr/batik/mlists/ batik users/archives/msg02806.html, 2003

[13] Simon Cox; Paul Daisey; Ron Lake; Clemens Portele; Arliss Whiteside, *Geography Markup Language (GML) ISO/TC 211/WG 4/PT 19136 OGC GML RWG*, Open GIS Consortium, Inc., 2004