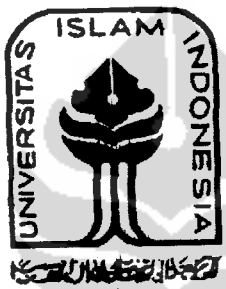
KONVERSI FILE XML KE DALAM TREE DAN MANIPULASINYA

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika



Oleh:

Nama : **Hendriarto Wigunawan**

No. Mahasiswa : **01 523 071**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PERNYATAAN KEASLIAN

HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,

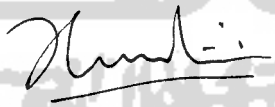
Nama : Hendriarto Wigunawan

No. Mahasiswa : 01 523 071

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, Januari 2007



(Hendriarto Wigunawan)

LEMBAR PENGESAHAN PENGUJI

KONVERSI FILE XML KE DALAM TREE DAN MANIPULASINYA

TUGAS AKHIR

Oleh:

Nama : Hendriarto Wigunawan

No. Mahasiswa : 01 523 071

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Januari 2007

Tim Penguji

Taufiq Hidayat ST., Mcs.
Ketua

Yudi Prayudi S.Si., M.Komp.,
Anggota I

Syarif Hidayat ST.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

Yudi Prayudi S.Si., M.Komp.

HALAMAN PERSEMBAHAN

Kupersembahkan kepada Bapak dan Ibuku yang telah memberikan nasehat, semangat, kasih sayang, fasilitas, dan selalu berdoa demi kebaikanku.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Segala puji bagi Allah swt. Tuhan seru sekalian alam. Berkat hidayah dan izin-Nya penyusun dapat menyelesaikan laporan penelitian yang berjudul "KONVERSI FILE XML KE DALAM TREE DAN MANIPULASINYA" sebagai salah satu syarat untuk memperoleh gelar sarjana.

Penulisan laporan ini bertujuan untuk mendokumentasikan beberapa hal yang berkaitan dengan penelitian yang dilakukan oleh penyusun, sehingga memungkinkan untuk dikembangkan lebih lanjut dikemudian hari.

Selama proses penyusunan berlangsung, penulis mendapat banyak dukungan dan bantuan dari berbagai pihak, sehingga penelitian ini dapat terlaksana sesuai dengan yang diharapkan. Oleh karena itu penulis ingin mengucapkan terimakasih yang sebesar-besarnya kepada:

1. Bapak Taufiq Hidayat, ST., MCS., yang telah membimbing penulis dari awal sampai akhir pelaksanaan tugas akhir.
2. Bapak Yudi Prayudi, Skomp., selaku Ketua Jurusan Teknik Informatika.
3. Bapak Fathul Wahid, ST., MSC., selaku dekan Fakultas Teknologi Industri.
4. Bapak Prof. Dr. Edy Suandi Hamid, M.Ec., selaku Rektor Universitas Islam Indonesia.
5. Mba Retno, Mas Heri, Brian, dan Intan yang memberikan suasana menyenangkan disaat melepas lelah.
6. Ita yang selalu memberikan dukungan, motivasi, dan kasih sayang.

7. Niko, Endang, Agung, Toshi, Sigit, Hendy, Almed, Rudy dan Vidi yang telah menjadi sahabatku dalam suka dan duka.
8. Teman-teman kos, Roby, Rifki, Gatot, Andes, Rivan, Agung, Priyo, Banu, dan Yudha yang selalu memberikan bantuan dan lingkungan yang nyaman..
9. Teman- teman angkatan 2001 Teknik Informatika yang menjadi sahabat dalam studiku.

Penulis menyadari bahwa masih banyak kekurangan dalam penelitian maupun penyusunan laporan ini, untuk itu penulis mengharapkan kritik dan saran yang membangun agar penlitian ini menjadi lebih berguna di masa yang akan datang.

Akhirnya, penulis berharap semoga penelitian ini dapat bermanfaat serta memperkaya khazanah ilmu pengetahuan di Indonesia.

Wassalam

Yogyakarta, Januari 2007

Penyusun

SARI

File *Extensible Markup Language (XML)* disimpan dalam bentuk teks, sehingga dapat ditulis, dibaca, dan diubah hanya dengan menggunakan editor teks biasa. Namun jika file tersebut semakin besar dan struktur didalamnya semakin kompleks, maka proses pembacaan dan manipulasi file XML akan menjadi sulit dilakukan.

Proses pembacaan dan manipulasi diharapkan akan menjadi lebih mudah dilakukan jika file XML divisualisasikan dalam struktur *tree*. Oleh karena itu, penelitian ini bertujuan untuk membangun sebuah aplikasi yang mampu melakukan konversi file XML ke dalam *tree* serta mempermudah proses manipulasi file tersebut.

Aplikasi didesain berdasarkan metode desain berorientasi pada aliran data. Sistem dideskripsikan sebagai proses atau kumpulan proses transformasi yang mengubah data masukan menjadi data keluaran. Sistem menerima masukan berupa file XML kemudian dikonversi ke dalam tree selanjutnya dikenai proses manipulasi berupa *cut*, *copy*, *insert*, *delete*, dan *edit*. Hasil keluaran adalah file XML yang telah termanipulasi.

Hasil ujicoba menunjukkan bahwa aplikasi hanya mampu menampilkan 10 jenis node ke dalam *tree* dan hanya mampu memanipulasi 7 jenis node. Sedangkan 2 jenis node yang lain hanya digunakan sebagai faktor dalam proses validasi.

Kata kunci : *tree, node, markup*

TAKARIR

ancestor	nenek moyang, sesuatu yang mendahului
attribute	sifat, perlengkapan
children	anak
document	dokumen
element	unsur, elemen
entity	entitas, sesuatu yang sungguh-sungguh ada
fragment	fragmen, pecahan, penggalan
markup language	bahasa untuk menandai bagaimana sebuah naskah direpresentasikan
node	titik cabang
notation	cara menulis, catatan
parent	orang tua, induk
prolog	kata pendahuluan, kata perkenalan
reference	referensi, rekomendasi
root	akar
sibling	saudara kandung
tag	label, tanda pengenal
tree	pohon
wellformed	berbentuk baik
valid	sah, absah, benar

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Semakin kompleksnya permasalahan yang dihadapi manusia menyebabkan kebutuhan akan informasi yang cepat dan akurat menjadi meningkat. Teknologi komputer dengan segala kecanggihan didalamnya mencoba menawarkan solusi untuk mengatasi kebutuhan akan informasi tersebut.

Perpindahan informasi menjadi mudah dilakukan setelah komputer mampu berkomunikasi dalam sebuah jaringan dan perpindahan ini semakin luas setelah ditemukannya internet yang mampu menghubungkan komputer di seluruh dunia. Melalui *web*, informasi dalam bentuk grafik, video, suara, maupun teks dapat disajikan melalui sebuah *browser*.

Hypertext Markup Language (HTML) dibuat dengan tujuan untuk mengatur tampilan di dalam *browser* tetapi tidak untuk menangani masalah struktur dari data di dalamnya. Oleh karena itu *World Wide Web Consortium (W3C)* sebagai badan yang mengawasi, mengatur, dan membuat standardisasi aplikasi *web*, mengembangkan bahasa lain yaitu *Extensible Markup Language (XML)*.

Mulai tahun 1998, XML ditetapkan menjadi sebuah standar dalam mendeskripsikan informasi di dalam aplikasi *web*. XML dapat berjalan di banyak *platform* sistem operasi seperti halnya HTML. Hal ini memberikan kemampuan dalam mengirim struktur sebuah

data dari berbagai macam aplikasi melalui jaringan internet untuk ditampilkan ke dalam desktop atau diproses lebih lanjut.

XML merupakan sebuah bahasa *markup*, sehingga didalamnya akan memuat banyak *tag-tag*. Berbeda dengan *tag* dalam HTML yang sudah tidak dapat dimodifikasi, *tag* XML dapat dibuat dan dimodifikasi oleh pengguna XML disesuaikan dengan permasalahan yang dihadapi. Struktur file XML seperti pohon (*tree*), yang terdiri dari beberapa jenis titik cabang (*node*).

File XML disimpan dalam bentuk teks, sehingga dapat ditulis, dibaca, dan diubah hanya dengan menggunakan editor teks biasa. Namun jika file tersebut semakin besar dan struktur didalamnya semakin kompleks, maka pembacaan akan menjadi sulit. Jika hal ini terjadi maka proses manipulasi dan validasi terhadap file XML menjadi semakin sulit dilakukan.

1.2 Rumusan Masalah

Berdasar pada kebutuhan terhadap program untuk pembacaan dan manipulasi file XML yang telah dijelaskan pada bagian latar belakang, maka dapat dirumuskan sebuah masalah yaitu, bagaimana membangun sebuah program yang mampu melakukan konversi file XML ke dalam *tree* serta melakukan manipulasi dan validasi terhadap file tersebut.

1.3 Batasan Masalah

Batasan masalah serta asumsi-asumsi dasar dalam pemecahan masalah yang digunakan dalam penelitian ini adalah sebagai berikut:

a. Versi XML yang dipakai adalah versi 1.0.

b. Deskripsi tipe dari *node* di dalam file XML mengacu pada *Document Type Definition (DTD)*.

c. Validasi dilakukan untuk mengetahui status *valid* dan *well-formed*.

d. *Parser* yang digunakan untuk menerjemahkan file XML ke dalam *tree* yaitu MSXML versi 4.0 yang merupakan implementasi dari standar *Document Object Model (DOM)*.

e. Aplikasi yang dikembangkan adalah berbasis Microsoft Windows dan perangkat pengembangan yang digunakan adalah Visual Basic 6.0.

1.4 Tujuan Penelitian

Berikut adalah tujuan yang diharapkan dapat tercapai dari pelaksanaan penelitian ini:

a. Tersediannya aplikasi berbasis *Windows* yang dapat melakukan konversi file XML ke dalam *tree* sehingga mempermudah dalam proses manipulasi isi dan struktur file tersebut.

b. Sistem yang dibangun dapat melakukan validasi terhadap file XML yang ter-manipulasi, sehingga file tersebut dapat digunakan untuk proses selanjutnya.

1.5 Manfaat Penelitian

Beberapa manfaat dari penelitian ini yang diharapkan dapat diambil adalah sebagai berikut:

a. Penelitian ini memberikan pengetahuan tentang struktur dari file XML versi 1.0.

b. Penggunaan standar *Document Object Model (DOM)* dan MSXML 4.0 dalam penelitian untuk menerjemahkaan file XML, dapat menambah wawasan mengenai cara kerja sebuah *parser*.

c. Penelitian ini memakai *Data Type Definition (DTD)* sebagai acuan validasi, sehingga dapat memberikan pengetahuan tentang cara pengecekan file XML diberbagai jenis aplikasi yang berbeda *platform* sistem operasi.

1.6 Metodologi Penelitian

1.6.1 Pengumpulan Data

Pengumpulan data serta referensi yang diperlukan dalam penelitian ini dilakukan melalui studi literatur berupa buku, jurnal ilmiah, makalah, dan artikel-artikel yang berhubungan dengan tema penelitian.

1.6.2 Pengembangan Perangkat Lunak

Pengembangan aplikasi konversi file XML ke dalam *tree* dibagi menjadi empat tahap mengacu pada standar pengembangan perangkat lunak:h

a. Analisis kebutuhan perangkat lunak, untuk pendefinisian ruang lingkup, kebutuhan sistem beserta fungsinya, unjuk kerja, dan antarmuka yang dibutuhkan.

b. Desain, untuk penentuan arsitektur perangkat lunak beserta antarmuka, struktur data, dan detail fungsionalnya.

c. Implementasi, merupakan tahap penerapan dari hasil analisis dan desain menjadi perangkat lunak yang dapat difungsikan sesuai tujuan pengembangannya.

d. Pengujian, untuk melakukan verifikasi dan validasi terhadap perangkat lunak yang dihasilkan.

1.7 Sistematika Penulisan

Bab I Pendahuluan, memberikan penjelasan tentang latar belakang penelitian, rumusan masalah yang dihadapi, batasan serta asumsi dari penyelesaian masalah yang telah dirumuskan sebelumnya, tujuan dan manfaat dari penelitian, metodologi penelitian yang digunakan, serta sistematika penulisan laporan.

Bab II Landasan Teori, memuat uraian tentang teori-teori yang berhubungan dengan penelitian. Teori yang dimaksud mencakup teori tentang pertukaran data dalam internet, file XML dan penggunaan *parser* DOM.

Bab III Analisis Kebutuhan, memberikan penjelasan tentang kebutuhan perangkat lunak yang akan dikembangkan selama penelitian, meliputi kebutuhan masukan, fungsi, keluaran, dan antarmuka yang dibutuhkan.

Bab IV Perancangan Perangkat Lunak, berisi uraian tentang arsitektur perangkat lunak berdasarkan kebutuhan yang diidentifikasi pada fase analisis, meliputi arsitektur, prosedur, antarmuka, dan tahapan-tahapan yang dilakukan untuk melakukan konversi file XML kedalam *tree* beserta manipulasinya.

Bab V Implementasi, memuat dokumentasi dari implementasi perangkat lunak yang dikembangkan, meliputi batasan pengembangan, implementasi struktur data, dan prosedur dalam kakas pemrograman Visual Basic 6.0, serta antarmuka yang dihasilkan.

Bab VI Analisis Kinerja, berisi dokumentasi selama pengujian perangkat lunak yang dikembangkan selama penelitian, meliputi pengujian data masukan dan penanganan kesalahan yang terjadidalam pengoperasian perangkat lunak.

Bab VII Simpulan dan Saran, berisi simpulan laporan tugas akhir dan saran-saran untuk pengembangan sistem sejenis yang lebih baik.

BAB II

LANDASAN TEORI

2.1 Pengertian dan Tujuan Utama XML

Extensible Markup Language (XML) merupakan sebuah bahasa *markup* yang menyediakan sebuah format untuk mendeskripsikan data yang terstruktur di dalam *web*[COR01]. Tujuan utama XML adalah untuk memenuhi kebutuhan akan sebuah format data yang dapat digunakan di berbagai aplikasi dan sistem operasi yang berbeda[COR98].

2.2 Penggunaan dan Kelebihan XML

XML memungkinkan adanya standar pendefinisian data yang disepakati oleh berbagai aplikasi dan *vendor*, sehingga XML dapat digunakan di dalam bidang *business-to-business (B2B)*, *web*, database, dan pengaturan dokumen. Berikut kelebihan dari penggunaan XML[COR 98]:

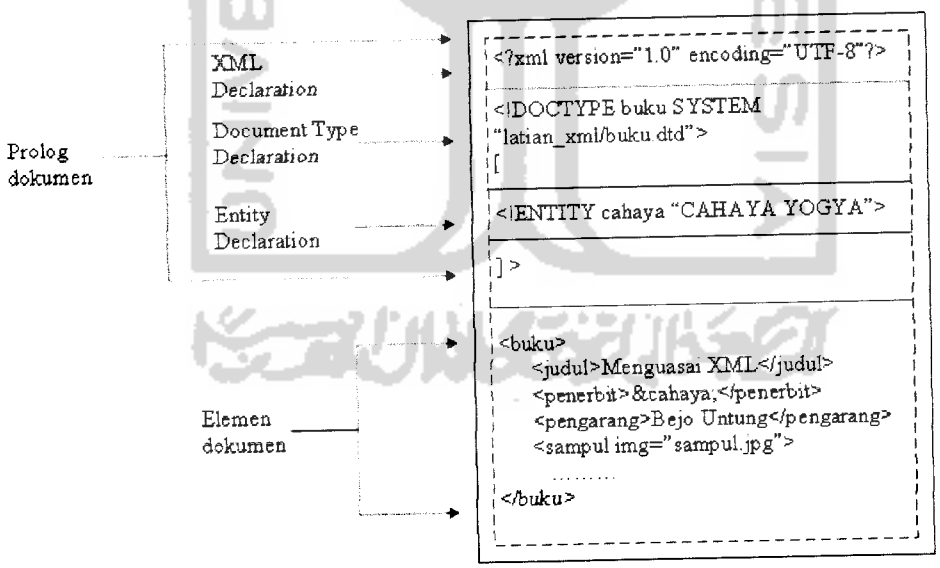
a. XML mengirimkan data untuk komputasi lokal
b. Memberikan perbedaan cara pandang pengguna terhadap data
c. Memungkinkan integrasi data yang terstruktur dari berbagai sumber
d. Mampu menggambarkan data dari berbagai macam aplikasi
e. Pencarian data yang lebih tepat sasaran
f. Menaikkan kecepatan melalui update bagian per bagian
g. Perpindahan data lebih mudah

2.3 Tag

Tag merupakan tanda atau pembatas yang diawali dengan tanda kurung sudut < dan diakhiri dengan tanda kurung sudut >[HAR04]. Berbeda dengan *tag* di dalam HTML yang hanya dapat dibuat oleh *World Wide Web Consortium* (W3C), *tag* XML dapat dibuat oleh penggunanya sehingga bersifat tidak terbatas (extensible)[HAR04].

2.4 Dokumen XML

Dokumen XML adalah konstruksi khusus yang didesain untuk menyimpan data[RAY03]. Dokumen tersebut dibagi menjadi beberapa bagian yang berbeda sesuai dengan fungsinya[RAY03]. Bagian-bagian dari dokumen ini bisa terdapat di dalam file yang sama, namun juga bisa terdapat di sistem dan jaringan yang berbeda[HAR04]. Berikut gambar bagian-bagian dokumen XML:



Gambar 2.1 Dokumen XML

Dokumen XML dibagi menjadi dua bagian, yang pertama adalah *prolog* dokumen, sedangkan yang kedua adalah elemen dokumen atau biasa disebut dengan elemen *root* [RAY03]. Elemen *root* mengandung seluruh elemen dan isi dokumen XML.

2.5 Prolog Dokumen

Prolog dokumen terletak di bagian atas dokumen sebelum *root* elemen. *Prolog* dokumen bersifat *optional,* jika tidak ditulis maka *parser* akan mengembalikannya ke nilai default. *Prolog* terdiri dari dua bagian, yang pertama adalah XML *declaration* berfungsi untuk pengaturan dasar, sedang yang kedua adalah *Document Type Declaration* untuk pengaturan tingkat lanjut.

2.5.1 XML Declaration

Bagian XML declaration berisi detil petunjuk kerja *parser* XML terhadap dokumen, diawali dengan *<?xml* dan diakhiri dengan *?>*[HAR04]. Berikut sintak XML declaration yang lengkap:

```
<?xml version ="1.0" encoding="UTF-8" standalone="YES"?>
```

Setiap parameter terdiri dari nama, tanda sama dengan (=), dan tanda petik. Parameter *version* harus terletak di urutan paling awal.

Version menunjukan versi dari XML yang digunakan. *Encoding* menunjukan jenis *Encoding* yang dipakai, nilai default-nya adalah UTF-8 [HAR04]. Sedangkan standalone mengindikasikan ada tidaknya deklarasi eksternal diluar dokumen. Nilai standalone ada dua, yaitu "NO" jika deklarasi hanya di dalam dokumen dan "YES" jika ada deklarasi di luar dokumen.

2.5.2 Document Type Declaration

Alasan penggunaan *Document Type Declaration* adalah memberi definisi pada entitas atau memberi nilai default pada atribut dan yang kedua adalah mendukung proses validasi pengecekan grammar[HAR04]. Berikut contoh sintak dari Document Type Declaration:

```
<!DOCTYPE buku SYSTEM "latian-xml/buku.dtd"
     [
     <!ENTITY penerbit "CAHAYA YOGYAKARTA">
]>
```

Identifier terdiri dari dua jenis yaitu *SYSTEM* dan *PUBLIC*. Identifier *SYSTEM* bersifat tidak tetap, bisa berubah kapan saja. Sedangkan *PUBLIC* bersifat lebih tetap, seperti spesifikasi elemen yang dibuat oleh W3C untuk HTML. *Entity Declaration* memberi nama pada bagian tertentu pada XML, kemudian dapat dipanggil kapan saja dan dapat diletakan di manapun. Berikut contohnya:

```
<buku>
     <judul>MENGUASAI XML</judul>
     <penerbit>&penerbit;</penerbit>
</buku>
```

2.6 Elemen

Elemen adalah penyusun file XML yang membagi file menjadi beberapa bagian, dan setiap bagian tersebut mempunyai makna tersendiri[RAY03]. *Root* elemen harus ada, karena dokumen tanpa data maka tidak bisa disebut dokumen. Nama elemen harus memenuhi beberapa aturan berikut:

a. Elemen terdiri dari satu karakter atau lebih

b. Karakter pertama harus berupa huruf atau *underscore* (_)

c. Tidak boleh mengandung spasi

d. Huruf besar dianggap berbeda dengan huruf kecil (*case sensitif*)

2.6.1 Well-formed atau Aturan Sintak

Dalam penulisan sintak XML haruslah sesuai dengan aturan yang dibuat oleh W3C, jika tidak maka tidak maka akan dianggap kesalahan sintak oleh *parser*[HAR04]. Penulisan sintak yang memenuhi kriteria dari W3C disebut dengan *Well-formed* yaitu harus memenuhi hal-hal berikut[WYK04]:

a. Setiap start *tag* harus diakhiri dengan end *tag*.

```
<judul>MENGUASAI XML</judul>
```

b. *Tag* kosong.

Karena setiap *tag* dalam XML harus ada penutupnya maka untuk *tag* yang tidak mempunyai isi (sehingga tidak mempunyai penutup) maka harus diawali dengan tanda < dan diakhiri dengan tanda /> seperti contoh berikut:

```
<image url="cover.jpg"/>
<judul_asli/>
```

c. Elemen tidak boleh saling mendahului.

Berikut sintak yang tidak tepat:

```
<penulis>Pangaribowo<judul>MENGUASAI XML</penulis></judul>
```

Seharusnya adalah:

```
<penulis>Pangaribowo<judul>MENGUASAI XML</judul></penulis>
```

d. Seluruh atribut harus berada dalam tanda petik.

```
<buku judul="MENGUASAI XML" harga="40.000">
```

e. Isi dari elemen XML dianggap sebagai data sehingga *white space* (spasi) tidak diabaikan.

Sehingga sintak:

```
<judul>MENGUASAI XML</judul>
```

Akan berbeda dengan:

```
<judul>MENGUASAI              XML</judul>
```

f. XML bersifat *case sensitif*.

Sintak:

```
<penulis>Pangaribowo</penulis>
```
Akan berbeda dengan:

```
<PENULIS>Pangaribowo</PENULIS>
```

2.6.2 Atribut

Di dalam start *tag* bisa ditempatkan informasi tambahan yang terdiri dari nama informasi dan nilainya. Informasi tambahan ini dinamakan atribut. Jumlah atribut di dalam elemen tidak terbatas[HAR04]. Berikut contohnya:

```
<judul penulis= "Pangaribowo" >Menguasai XML</judul>
```
Selain sebagai informasi tambahan, atribut juga bisa digunakan untuk membedakan elemen yang mempunyai nama sama. Berikut contohnya:

```
<buku id="fiksi">...</buku>
```

Akan berbeda dengan

```
<buku id="nonfiksi">...</buku>
```

2.7 Tree

Tree / pohon merupakan penggambaran struktur suatu data dari atas ke bawah[RAY03]. Beikut bagian-bagian dari sebuah *tree* XML:

a. Elemen paling luar (*root*) bisa diibaratkan seperti sebuah batang pohon

b. Elemen yang berada di bawah *root* diibaratkan sebagai ranting / sub *tree*

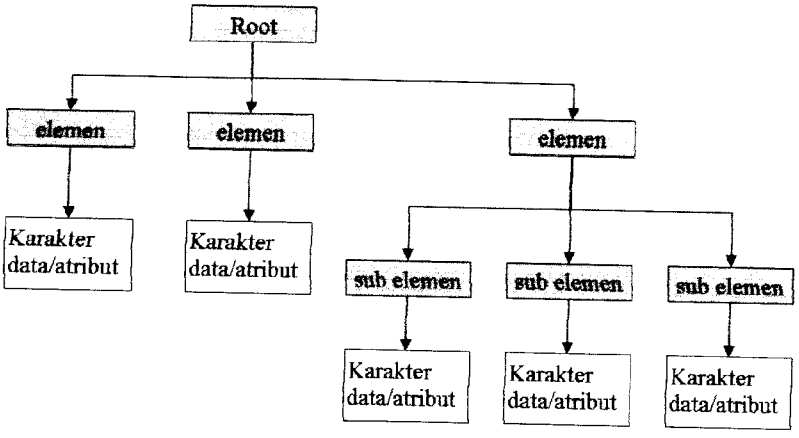
c. Elemen di dalam elemen / sub elemen dianggap juga seperti ranting

d. Atribut dan data karakter yang diibaratkan sebagai daun

e. Setiap titik di dalam *tree* (elemen, teks, atau yang lainnya) disebut *node*

f. Kumpulan dari *tree* disebut dengan *grooves* (hutan kecil).

Gambar 2.2 adalah sebuah *tree*:



Gambar 2.2 *Tree*

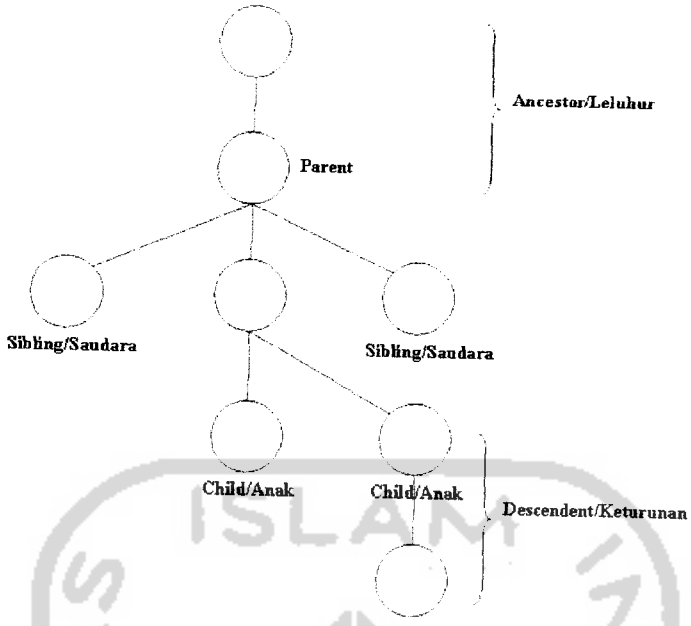
Proses parsing akan dilakukan dari elemen *root* menuju elemen yang lain kemudian akan diteruskan menuju karakter data dan atribut.

2.7.1 Silsilah Tree

Untuk menjelaskan "hubungan" antara satu elemen dengan elemen yang lain maka dapat digunakan pohon silsilah seperti pada silsilah keluarga[RAY03]. Elemen dapat diibaratkan sebagai anggota dari sebuah keluarga yang bisa memiliki keturunan. Dengan ketentuan sebagai berikut[RAY03]:

a. *Root* dalam silsilah bisa diibaratkan sebagai orang paling tua dalam keluarga elemen disebut sebagai *ancestor* atau nenek moyang
b. Keturunan / descendant adalah elemen ada di dalam *root*
c. Elemen lain yang mempunyai isi berupa elemen disebut orang tua (*parent*)
d. Elemen yang memiliki *parent* disebut dengan anak (*children*)
e. Elemen yang mempunyai kedudukan sejajar dinamakan dengan saudara (*sibling*)

Gambar 2.3 adalah silsilah *tree*:



Gambar 2.3 Silsilah *TREE*

2.7.2 Fungsi Tree

Setiap dokumen XML hanya dapat memiliki satu buah *tree*, karena ciri yang unik ini maka *tree* dapat digunakan sebagai pengidentifikasi atau pembuat beda antara satu dokumen dengan dokumen yang lain. Sifatnya yang tidak membingungkan membuat XML sangat berguna untuk menyimpan data. Selain itu *tree* mempunyai beberapa kegunaan lain, yaitu[RAY03]:

a. *Tree* yang mempunyai batang, ranting, dan daun, membuatnya mudah dirunut sehingga penelusuran dokumen akan lebih mudah.

b. Model *tree* juga sangat berguna karena menggambarkan bagaimana XML disimpan dalam memori komputer. Setiap elemen dan bagian dari teks dikumpulkan dalam sebuah *cell* dengan pointer antara *children* dan *parent*.

c. Sistem *tree* yang memisahkan bagian per bagian dengan jelas juga dapat memudahkan *developer* untuk memindahkan dan mencari teks.

2.8 Document Type Definition (DTD)

Document Type Definition (DTD) merupakan salah satu rekomendasi dari W3C untuk validasi XML[HAR04]. Berikut cara DTD mendefinisikan tipe dokumen:

a. Mendeklarasikan elemen yang diperbolehkan. Nama elemen yang tidak terdapat di dalam DTD tidak diizinkan untuk dipergunakan.

b. Menjelaskan bentuk isi yang boleh dimasukan ke dalam sebuah elemen.

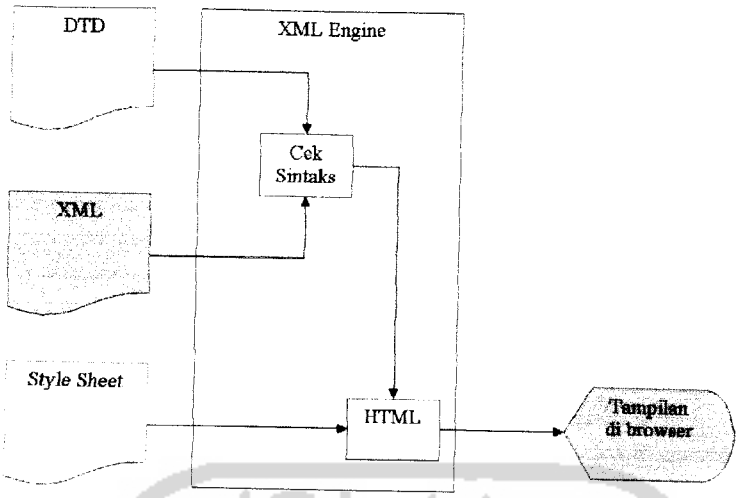
c. Mendeklarasikan jenis atribut yang diizinkan dalam tiap elemen.

d. Menyediakan beberapa variasi mekanisme untuk mengatur dokumen.

DTD dapat disimpan dalam file terpisah dari dokumen (eksternal) dengan ekstensi *.dtd* atau ditulis langsung dalam satu file dengan dokumen (internal).

2.9 Mekanisme Kerja XML

Dokumen XML yang telah dibuat dapat divalidasi dengan menggunakan DTD, sehingga dihasilkan output yang sesuai dengan aturan-aturan yang ditetapkan dalam DTD. Style-sheet seperti *Extensible Style Language* (XSL) atau *Cascade Style Sheets* (CSS) dibutuhkan untuk menjelaskan bagaimana cara menampilkan bagian-bagian XML sehingga dapat digabungkan dengan HTML dan kemudian ditampilkan dalam *browser*[COR01]. Gambar 2.4 menunjukkan cara kerja XML:



Gambar 2.4 Cara Kerja XML

2.10 Validasi Valid

Dokumen XML akan dianggap valid jika memenuhi beberapa kriteria berikut ini:

a. Harus *well formed*

b. Harus mempunyai *Document Type Declaration*

c. Mempunyai sebuah *root* elemen yang dirujuk oleh Document Type Declaration

d. Sesuai dengan ketentuan di dalam *Document Type Definition* (DTD).

2.11 Parser

Parser adalah sebuah aplikasi yang mampu membagi data menjadi bagian-bagian yang lebih kecil yang dapat didefinisikan oleh aturan-aturan suatu bahasa pemrograman[COR01]. Keberadaan *parser* menjadikan aplikasi yang lain mampu berbuat sesuatu berdasarkan informasi yang disediakan oleh *parser* tersebut.

2.12 Document Object Model (DOM) dan MSXML 4.0

DOM merupakan sebuah standar untuk serangkaian perintah yang harus dimiliki oleh sebuah *parser* untuk mengakses dokumen XML dan HTML dari sebuah aplikasi[RAN99]. DOM untuk XML adalah sebuah model objek yang memperlihatkan isi dari sebuah dokumen XML. Spesifikasi W3C tentang DOM menjelaskan bahwa sebuah DOM haruslah mampu menunjukkan properti, metode dan kejadian dari XML.

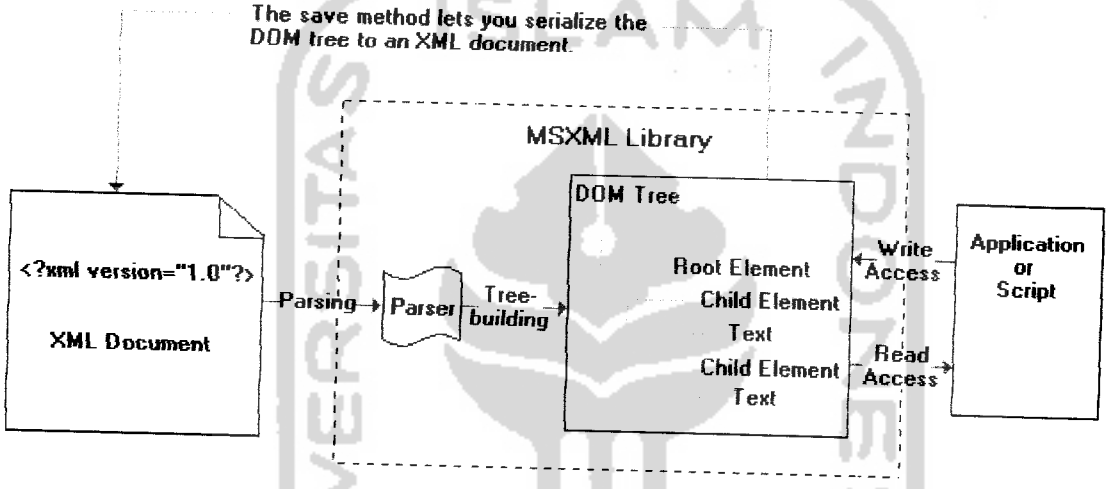
DOM mendeskripsikan XML sebagai *tree* yang terdiri dari beberapa jenis node. Tabel berikut menjelaskan jenis node yang terdapat di dalam DOM:

Tabel 2.1 Jenis Node dalam DOM [LAN99]

Jenis Node	Contoh
NODE_ELEMENT	<ELEMENT> ... </ELEMENT>
NODE_ATTRIBUTE	<ELEMENT ATTRIBUTE="This is a node attribute" />
NODE_TEXT	<ELEMENT> This is a node text. </ELEMENT>
NODE_CDATA_SECTION	<ELEMENT><![CDATA[a CDATA section is used to escape a block of text that would otherwise be recognized as *markup*.]]> </ELEMENT>
NODE_ENTITY_REFERENCE	<ELEMENT> &ent;</ELEMENT>
NODE_ENTITY	<!ENTITY %textgizmo "fontgizmo">
NODE_PROCESSING_INSTRUCTION	<?xml version="1.0" standalone="yes" ?>
NODE_COMMENT	<!--This is a comment -->
NODE_DOCUMENT	<XMLROOTELEMENT> ... (entire xml tree </XMLROOTELEMENT>
NODE_DOCUMENT_TYPE	<!DOCTYPE INVOICE SYSTEM "http://www.xml.com/product.dtd">

NODE_DOCUMENT_FRAGMENT	<ELEMENT> ... (subtree) ... </ELEMENT>
NODE_NOTATION	<!NOTATION gif SYSTEM "viewer.exe">

Microsoft XML parser (MSXML) versi 4.0 merupakan salah satu parser yang merupakan implementasi dari DOM yang dibuat oleh Microsoft. Berikut merupakan cara kerja MSXML 4.0.



Gambar 2.5 Cara Kerja MSXML 4.0

File XML pertama kali akan di-parsing oleh parser MSXML sehingga secara logika akan terbentuk tree. Untuk merubahnya dalam bentuk fisik maka perllu digunakan aplikasi tambahan. Melalui aplikasi tersebut file XML dapat dimanipulasi.

BAB III

ANALISIS KEBUTUHAN

3.1 Metode Analisis

Metode yang digunakan untuk analisa kebutuhan masukan, proses, dan keluaran adalah metode analisis terstruktur (*structured analysis*). Metode ini berorientasi kepada aliran data yang masuk dan keluar dari sistem. Metode ini menganut sistem dekomposisi, yaitu menggambarkan terlebih dahulu sistem secara keseluruhan sebagai tingkat tertinggi kemudian memecahnya menjadi bagian-bagian yang lebih terinci.

3.2 Hasil Analisis

Hasil dari analisis kebutuhan adalah kebutuhan perangkat lunak yang hendak dibangun. Kebutuhan yang dimaksud adalah kebutuhan masukan, keluaran, kebutuhan antarmuka, dan kebutuhan proses atau fungsi. Perangkat lunak yang dihasilkan dalam penelitian ini bernama *Sistem Konversi dan Manipulasi File XML versi 1.0*.

3.2.1 Masukan (Input)

Masukan yang dibutuhkan oleh perangkat lunak yang dibangun dalam penelitian ini adalah:

a. File *Extensible Markup Language* (XML) dengan format (*.xml), sebagai masukan yang akan dikenai proses konversi dan validasi, contoh:

```
<?xml version="1.0" standalone="no" encoding="UTF-16"?>
<!DOCTYPE buku SYSTEM "C:\buku.dtd"
     [
```

```
    <!ENTITY LPI "LEMBAGA PUSTAKA ILMU">
]>
<buku>
    <judul sampul="buku XML.gif">Memahami XML</judul>
    <penerbit>
    <nama>&LPI;</nama>
  <alamat>JAKARTA</alamat>
    </penerbit>
    <tgl_terbit>12-5-2004</tgl_terbit>
</buku>
```

b. File *Document Type Definition* (DTD) dengan format (*.dtd), berisi aturan–aturan yang digunakan untuk proses validasi file XML, contoh:

```
<!ELEMENT buku (judul*)>
<!ELEMENT judul (penerbit, tgl_terbit)>
<!ELEMENT penerbit (nama, alamat) >
<!ELEMENT nama (#PCDATA)>
<!ELEMENT alamat (#PCDATA)>
<!ELEMENT tgl_terbit (#PCDATA)>
<!ATTLIST judul sampul CDATA #IMPLIED>
```

c. file *Cascade Style Sheets* (CSS) dengan format (*.css), dibutuhkan untuk menjelaskan bagaimana cara menampilkan bagian-bagian XML, contoh:

```
judul {
     display: block;
     font-family: serif;
     font-size: 16pt;
}
penerbit {
     display : block;
     font-size: 12pt;
}
tgl_terbit{
     display : block;
     font-size: 12pt;
}
```

3.2.2 Proses

Terdapat tiga buah proses utama yang dibutuhkan oleh perangkat lunak, yaitu konversi file XML ke dalam *tree*, manipulasi struktur *tree*, kemudian konversi *tree* ke dalam file XML.

3.2.2.1 Konversi File XML ke dalam Tree

Fungsi konversi file XML ke dalam *tree* adalah yang pertama kali dijalankan dalam perangkat lunak. Tujuan dari proses ini adalah untuk mendapatkan deskripsi atau gambaran keseluruhan dari struktur file XML masukan. Setiap bagian dari file XML tersebut akan dibedakan menurut jenisnya sesuai dengan aturan yang ada di dalam DOM.

3.2.2.2 Manipulasi Struktur Tree

Proses ini merupakan kelanjutan dari proses sebelumnya dimana file sudah direpresentasikan dalam sebuah *tree*. Proses manipulasi meliputi penambahan, penghapusan, dan penambahan struktur *tree*. Berikut proses yang dapat dilakukan dalam memanipulasi struktur *tree*:

a. Manipulasi node element
b. Manipulasi node attribute
c. Manipulasi node text
d. Manipulasi node cdata
e. Manipulasi node processing instruction
f. Manipulasi node comment

Selain itu terdapat beberapa proses pendukung dalam pemanipulasian *tree* yaitu:

a. Validasi file XML untuk wellformed
b. Validasi file XML dengan DTD yang ditunjuk
c. Penggabungan dengan stylesheet CSS
d. Menampilkan file XML di dalam *Browser*

3.2.2.3 Konversi Tree ke dalam File XML

Proses konversi *tree* ke dalam file XML merupakan proses dimana *tree* telah dimodifikasi. File XML tersebut selanjutnya bisa digunakan untuk kebutuhan *web* atau aplikasi lainnya.

3.2.3 Keluaran (Output)

Keluaran yang dihasilkan oleh aplikasi dalam penelitian ini dibagi menjadi dua bagian, yaitu *tree* dan file XML yang telah dimanipulasi.

3.2.4 Antarmuka Sistem

Antarmuka merupakan penghubung antara pengguna dengan sistem. Agar pengguna dapat mengeksplorasi sistem dengan mudah maka diperlukan sebuah antarmuka yang tidak membingungkan (*user friendly*). Corak interaksi yang digunakan dalam penelitian ini adalah WIMP atau *Windows, Icons, Menus, dan Pointers*. Corak ini biasa digunakan untuk aplikasi berbasis *Windows* (*Windows-based application*).

BAB IV

PERANCANGAN PERANGKAT LUNAK

4.1 Metode Perancangan

Setelah diperoleh spesifikasi kebutuhan aplikasi perangkat lunak pada tahap analisis kebutuhan, langkah selanjutnya adalah perancangan perangkat lunak. Tahap ini merupakan langkah awal untuk implementasi dan pengembangan perangkat lunak. Dalam rekayasa perangkat lunak, tahap perancangan merupakan tahap yang sangat menentukan kualitas perangkat lunak yang dihasilkan. Oleh karena itu diusahakan tahap perancangan ini dibuat dan didokumentasikan sebaik mungkin agar perangkat lunak yang dihasilkan sesuai dengan yang diharapkan.

Aplikasi perangkat lunak ini didesain berdasarkan metode desain berorientasi pada aliran data. Sistem dideskripsikan sebagai proses atau kumpulan proses transformasi yang mengubah data masukan menjadi data keluaran. Diagram aliran data digunakan untuk menggambarkan aliran informasi dan transformasi data yang terjadi di dalam sistem.

4.2 Hasil Perancangan

Bab ini berisi dokumentasi hasil perancangan perangkat lunak yang dibangun dalam penelitian. Dokumentasi terdiri dari model data yang hendak digunakan, desain arsitektur perangkat lunak, dan desain antarmuka aplikasi yang hendak dibangun.

4.3 Desain Data

Struktur XML yang kompleks tidak memungkinkan XML diterjemahkan secara langsung tanpa menggunakan program bantu. Dibutuhkan sebuah *parser* untuk menerjemahkan struktur XML yang kompleks tersebut menjadi data yang mudah dipahami.

Document Object Models(DOM) merupakan salah satu dari model dalam mem-parsing XML. Dalam pembuatan perangkat lunak ini akan digunakan sebuah *parser* XML yang merupakan implementasi dari DOM yaitu *Microsoft's XML parser (MSXML)* versi 4.0. Tabel 4.1 menerangkan nama-nama node di dalam DOM beserta deskripsi node tersebut:

Tabel 4.1 Jenis node beserta deskripsinya

No	Nama Node	Nama Dlm Tree	Isi Node	Children
1	ELEMENT	Nama dari elemen	null	Element, text, comment, cdata, entity reference
2	ATTRIBUTE	Nama dari atribut	teks	Tidak ada
3	TEXT	#text	teks	Tidak ada
4	CDATA_SECTION	#cdata	Teks cdata	Tidak ada
5	ENTITY_REFERENCE	nama entity reference	Null	Element, text, comment, cdata, processing instruction
6	ENTITY	Nama dari entity	Null	Element, text, comment, cdata, entity reference
7	PROCESSING_INSTRUCTION	Nama dari target	Teks processing instruction	Tidak ada

8	COMMENT_	#comment	Teks komentar	Tidak ada
9	DOCUMENT	#document	null	
10	DOCUMENT_TYPE	Nama dari document type	null	Tidak ada
11	DOCUMENT_FRAGMENT	#document-fragmen	null	Element, text, comment, cdata, entity reference, processing instruction
12	NOTATION	Nama dari notasi	null	Tidak ada

Node-node di atas mempunyai nama dan karakteristik yang berbeda, oleh karena itu penanganan masing-masing node juga tidak sama.

4.3.1 Penggunaan Data

Hampir seluruh node dalam XML dapat diakses dan ditampilkan menggunakan MSXML dalam sebuah *tree*. Penggunaan node dalam MSXML akan mengacu pada property dan metode yang dikenakan pada node-node tersebut. Sehingga proses menampilkan dan manipulasi node dalam XML akan sangat bergantung pada property dan metode yang dimiliki masing-masing node dalam MSXML.

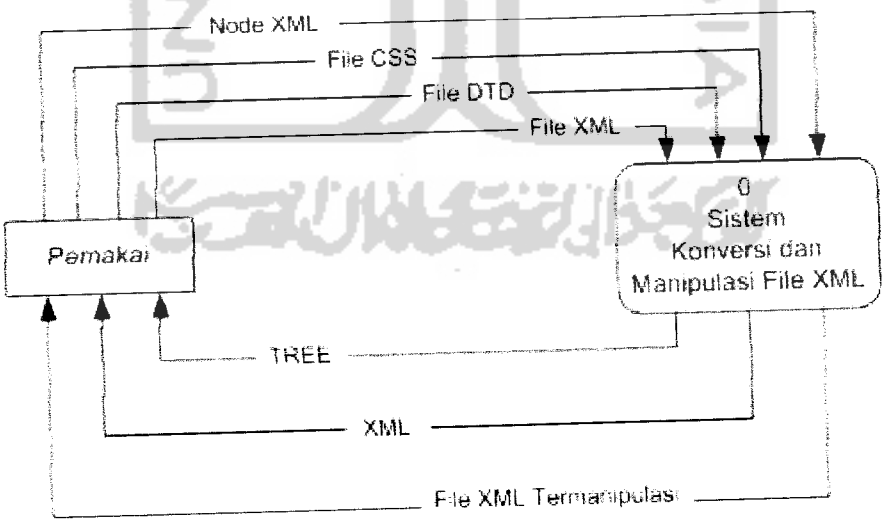
Berbeda dengan proses untuk menampilkan node, proses manipulasi node tidak didukung penuh oleh MSXML, hanya beberapa node tertentu yang dapat dimanipulasi di dalam sebuah *tree*, yaitu node *element, attribute, text, cdata, comment,* dan *processing instruction*. Sedangkan node yang lain hanya dapat dibaca atau sebagai acuan untuk proses validasi.

4.4 Desain Arsitektur Perangkat Lunak

Diagram aliran data / *data flow diagram* (DFD) adalah sebuah teknik grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari input menjadi output[PRE02a]. Agar transformasi data dapat dilakukan maka dibutuhkan beberapa proses transformasi. Berikut diagram alir data yang menjelaskan proses transformasi yang terjadi di dalam aplikasi konversi file XML ke dalam *tree* yang hendak dibangun. Notasi diagram aliran data yang digunakan adalah notasi Gane-Sarson.

4.4.1 DFD level konteks (0)

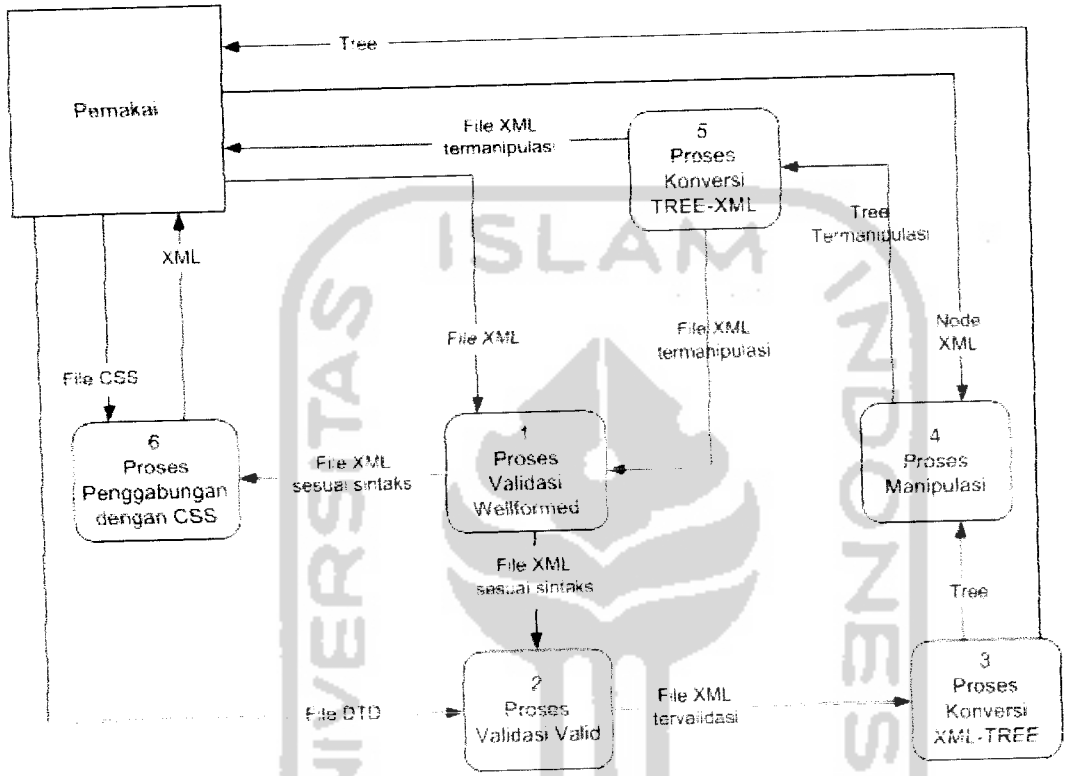
Dalam gambar 4.1 diilustrasikan abstraksi dari sistem secara keseluruhan. Entitas eksternal yang berhubungan dengan sistem dan pemakai. Masukan yang diterima sistem berupa file XML setelah diproses akan menghasilkan *tree*, file XML termanipulasi, dan tampilan XML dalam *browser* .



Gambar 4.1 DFD level konteks (0)

4.4.2 DFD level satu (1)

Gambar 4.2 mengilustrasikan uraian tentang proses-proses transformasi yang ada di dalam Aplikasi Konversi dan Manipulasi file XML.



Gambar 4.2 DFD level satu (1)

Sistem konversi dan manipulasi file XML terdiri dari beberapa proses, berikut uraian dari proses-proses tersebut:

1. Proses validasi wellformed mempunyai tujuan agar file XML yang masuk telah memenuhi kriteria sintaks XML yang baik dan benar.
2. Proses validasi valid menerima masukan berupa file XML yang wellformed dan file DTD. Setiap elemen file XML yang mempunyai DTD haruslah sesuai dengan ketentuan yang tertulis di dalam DTD. Jika ada elemen yang tidak sesuai dengan file DTD maka akan dianggap tidak valid.

3. Proses konversi XML-Tree hanya dapat menerima masukan berupa file XML yang telah memiliki status wellfomed dan status valid (jika mempunyai DTD). Konversi dilakukan untuk menghasilkan visualisasi struktur *tree* dari node-node dalam file XML.

4. Proses manipulasi bertujuan untuk menghapus, menambah, ataupun mengubah node di dalam *tree*. Proses ini tidak berhubungan langsung dengan file XML melainkan berhubungan dengan tampilan *tree* yang merupakan representasi dari struktur file XML. Sehingga proses manipulasi dapat berpengaruh setelah mendapatkan proses konversi ke dalam file XML lagi.

5. Proses konversi Tree-XML akan menjadikan manipulasi *tree* diimplementasikan pada file XML yang sesungguhnya. Untuk memastikan file XML hasil manipulasi sesuai dengan aturan sintaks maupun valid, maka dilakukan kembali proses validasi wellformed dan validasi valid. Jika terdapat kesalahan maka file tersebut dapat dikoreksi/dimanipulasi kembali.

6. Proses penggabungan dengan file CSS berfungsi mengambil file CSS yang alamatnya ditunjuk oleh processing instruction file XML. Kemudian menggabungkan file CSS tersebut dengan file XML, sehingga setiap node dalam file XML memiliki format untuk ditampilkan di dalam *browser*.

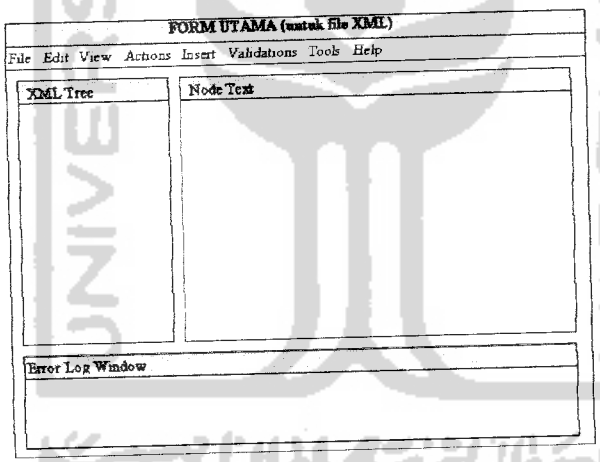
4.5 Desain Antarmuka

Antarmuka pemakai dengan sistem merupakan bagian yang tidak terpisahkan dari sebuah aplikasi perangkat lunak. Antarmuka digunakan sebagai media interaksi antara pemakai dengan sistem. Dengan adanya desain antarmuka diharapkan antarmuka yang dihasilkan bersifat ramah, efektif, dan efisien.

Desain antarmuka yang baik memungkinkan pemakai untuk memanfaatkan fungsi-fungsi yang disediakan oleh perangkat lunak dengan mudah. Sehingga pemakai dapat mengoptimalkan fungsi-fungsi tersebut tanpa merasa cepat lelah dan bosan. Berikut ini adalah perancangan tampilan untuk fungsi-fungsi dasar yang tersedia di dalam perangkat lunak yang dihasilkan.

4.5.1 Perancangan tampilan Form Utama untuk file XML

Form Tree merupakan media interaksi yang digunakan untuk menampilkan hasil konversi dari file XML ke dalam struktur *tree*. Node-node file XML tersebut akan direpresentasikan menjadi node-node dalam *tree* dengan simbol yang berbeda sesuai dengan jenisnya. Gambar 4.3 mengilustrasikan perancangan Form Tree.



Gambar 4.3 Form Utama untuk file XML

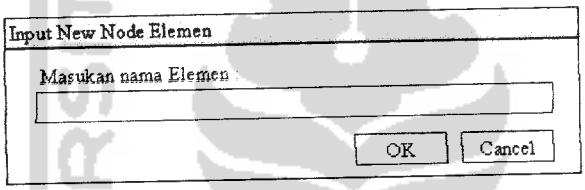
Hampir seluruh perintah dalam form utama dilakukan melalui menu form dan *pop-up* menu, sehingga form terlihat lebih luas. *Tree* XML akan ditampilkan pada halaman form sisi kiri. Isi dari node teks akan ditampilkan pada halaman di sebelah kiri satu barus dengan node teks. Halaman *Error Log* berfungsi untuk menampilkan kesalahan-kesalahan yang terjadi pada penggunaan aplikasi.

4.5.2 Perancangan tampilan Form Edit dan Insert Node

Node di dalam file XML dapat mempunyai sifat yang sama namun juga dapat mempunyai sifat yang berbeda, sehingga perancangan Form Edit dan Insert untuk masing masing node akan berbeda. Bentuk form edit untuk suatu node akan sama dengan form inputnya. Perbedaannya hanya terletak pada judul form tersebut.

a. Form Input / Edit Node Elemen

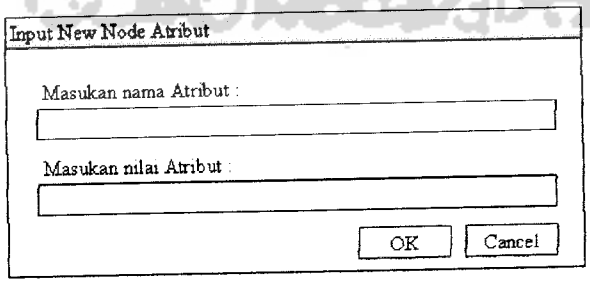
Form ini berfungsi untuk memasukan node elemen yang baru ke dalam *tree* atau mengubah node elemen. Gambar 4.4 merupakan gambar rancangan dari form input elemen.



Gambar 4.4 Form Input Node Elemen

b. Form Input / Edit Node Atribut

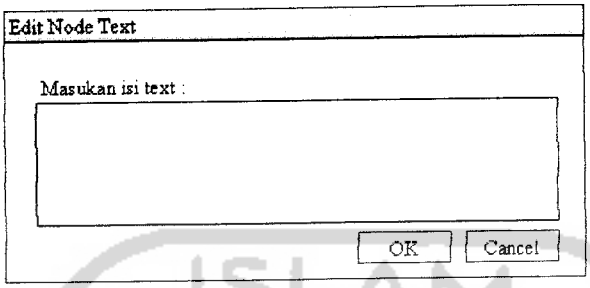
Form ini berfungsi untuk memasukan node atribut yang baru ke dalam *tree* atau mengubah node atribut. Gambar 4.5 merupakan gambar rancangan dari form input atribut.



Gambar 4.5 Form Input Node Atribut

c . Form Input / Edit Node Text

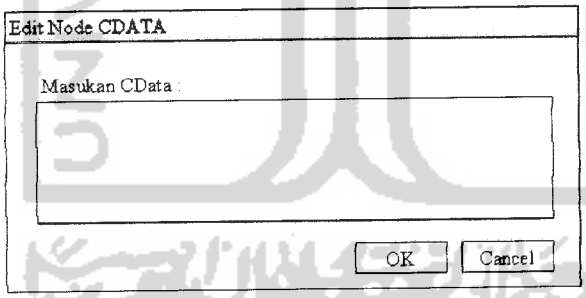
Form ini berfungsi untuk memasukan node text yang baru ke dalam *tree* atau mengubah node text. Gambar 4.6 merupakan gambar rancangan dari form input text.



Gambar 4.6 Form Input Node Text

d. Form Input / Edit Node CDATA

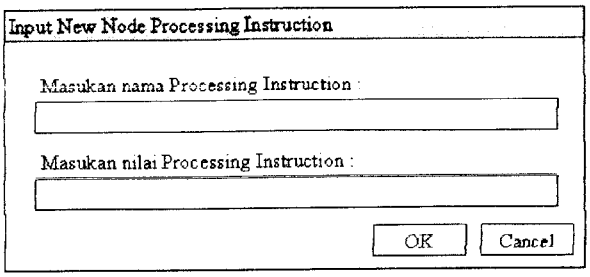
Form ini berfungsi untuk memasukan node CDATA yang baru ke dalam *tree* atau mengubah node CDATA. Gambar 4.7 merupakan gambar rancangan dari form input CDATA.



Gambar 4.7 Form Input Node CDATA

e. Form Input / Edit Node Processing Instruction

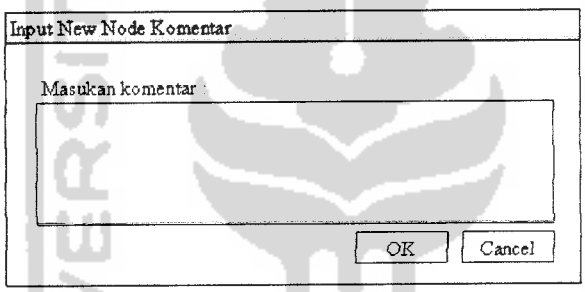
Form ini berfungsi untuk memasukan node processing instruction yang baru ke dalam *tree* atau mengubah node processing instruction. Gambar 4.8 merupakan gambar rancangan dari form input Processing Instruction.



Gambar 4.8 Form Input Node Processing Instruction

f. Form Input / Edit Node Komentar

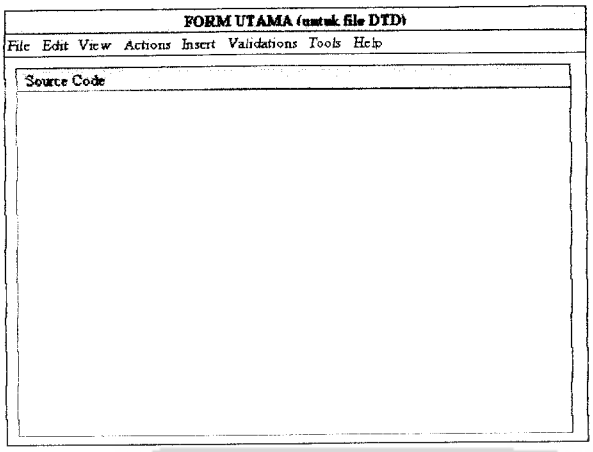
Form ini berfungsi untuk memasukan node komentar yang baru ke dalam *tree* atau mengubah node komentar. Gambar 4.5 merupakan gambar rancangan dari form input komentar.



Gambar 4.9 Form Input Node Komentar

4.5.3 Perancangan tampilan untuk file DTD

File DTD digunakan oleh File XML sebagai *validator* atau sebagai rujukan. Sehingga aplikasi tidak mengkonversi file DTD menjadi bentuk *tree*. Pembuatan file XML yang valid harus sesuai dengan DTD-nya, oleh karena itu sistem yang akan dibuat dilengkapi dengan fasilitas untuk melihat file source DTD dan memanipulasi isinya. Gambar 4.10 mengilustrasikan rancangan form utama untuk file DTD.

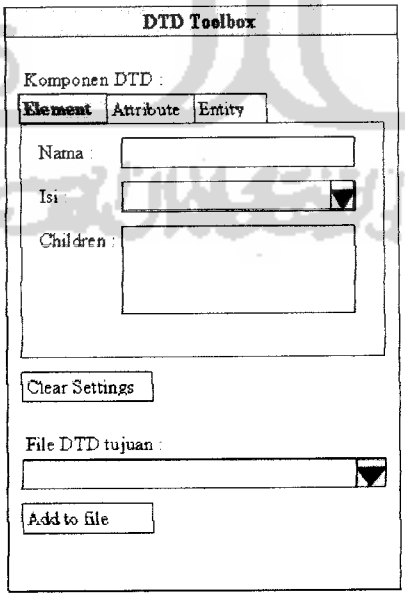


Gambar 4.10 Form Utama untuk file DTD

4.5.4 Perancangan tampilan untuk form pembuat DTD sederhana

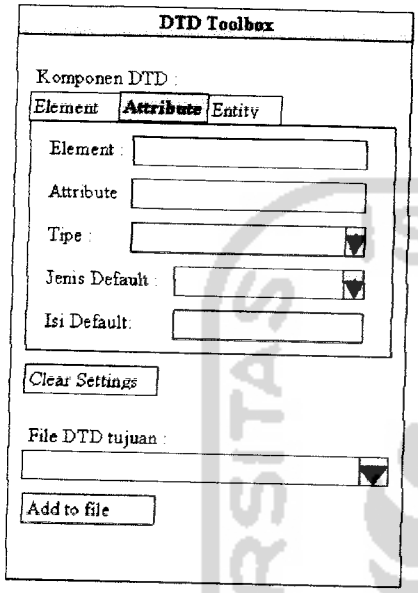
Source code DTD yang ditampilkan dalam form utama dapat dimanipulasi dengan diketik secara langsung. Form yang akan dibuat ini berguna untuk membantu pembuatan file DTD, dengan cara mengelompokan masing-masing fungsi sesuai dengan node yang akan dibuat.

a. Form untuk membantu pembuatan Elemen (gambar 4.11) dalam DTD

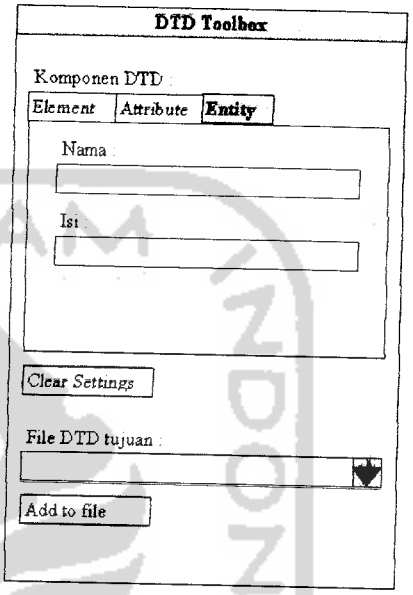


Gambar 4.11 Form Elemen DTD

b. Form untuk membantu pembuatan Atribut (gambar 4.12) dan Entity (gambar 4.13) dalam DTD



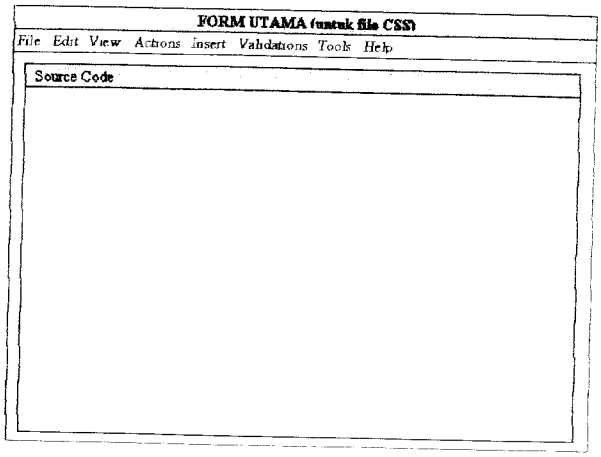
Gambar 4.12 Form Atribut DTD



Gambar 4.13 DFD Form Entity DTD

4.5.5 Perancangan tampilan untuk file CSS

Form untuk source file CSS diperlukan untuk memodifikasi atau untuk membaca file CSS yang diminta oleh file XML di dalam node Processing Instruction. Gambar 4.14 mengilustrasikan rancangan form utama untuk file CSS.

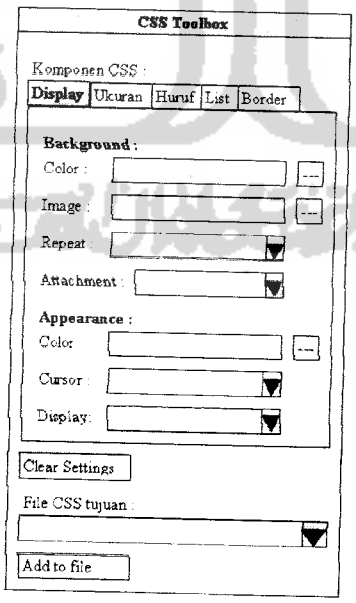


Gambar 4.14 Form Utama untuk file CSS

4.5.6 Perancangan tampilan untuk form pembuat CSS sederhana

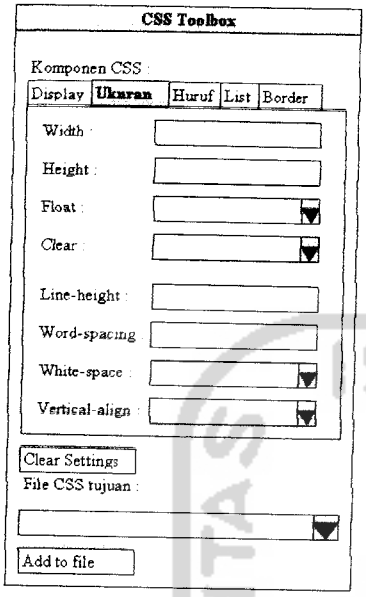
Source code CSS yang ditampilkan dalam form utama dapat dimanipulasi dengan diketik secara langsung. Form yang akan dibuat ini berguna untuk membantu pembuatan file CSS, dengan cara mengelompokan masing-masing fungsi sesuai dengan format tampilan node yang akan dibuat.

a. Form untuk membantu setting Display dalam CSS seperti terlihat pada gambar 4.15

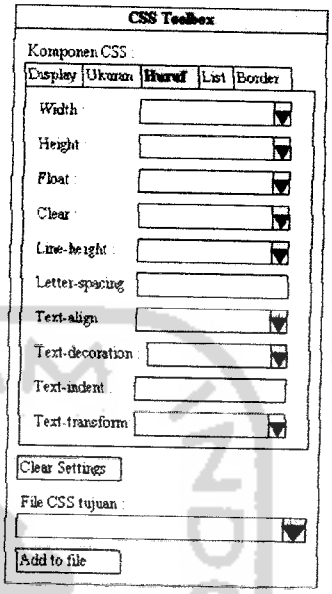


Gambar 4.15 Form Display CSS

b. Form untuk setting Ukuran (gambar 4.16) dan Huruf (gambar 4.17) CSS

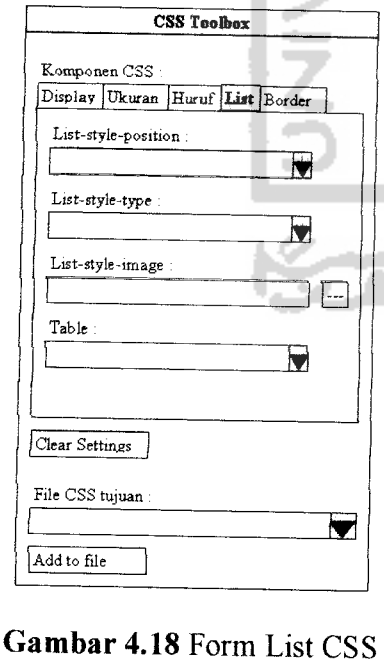


Gambar 4.16 Form Ukuran CSS

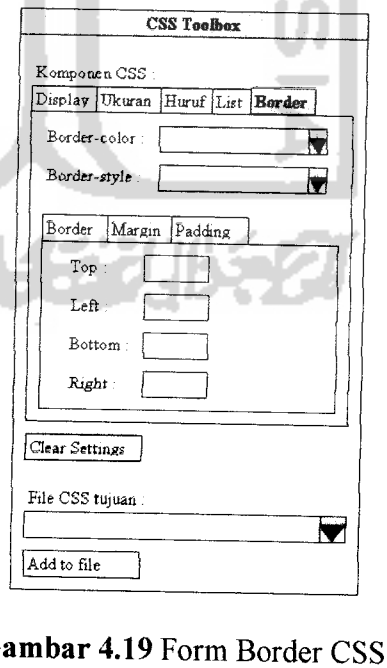


Gambar 4.17 Form Huruf CSS

c. Form untuk setting List (gambar 4.18) dan Border (gambar 4.19) dalam CSS



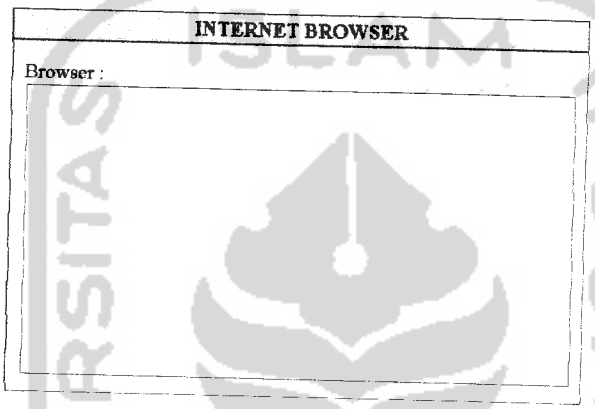
Gambar 4.18 Form List CSS



Gambar 4.19 Form Border CSS

4.5.7 Perancangan tampilan untuk Browser XML

Form Browser Internet dapat menampilkan source file XML yang telah lulus validasi wellformed. Jika di dalam file XML tersebut terdapat processing instruction yang merujuk pada file CSS, maka file XML akan ditampilkan dengan format file CSS tersebut. Gambar 4.20 mengilustrasikan rancangan form untuk *browser* internet.



Gambar 4.20 Internet *Browser*

BAB V

IMPLEMENTASI

5.1 Batasan Implementasi

Berikut batasan-batasan implementasi perangkat lunak dari sisi hardware dan software yang digunakan dalam pengembangan perangkat lunak serta spesifikasi hardware maupun software yang diperlukan untuk menjalankan perangkat lunak yang dihasilkan dalam penelitian ini.

5.1.1 Kebutuhan perangkat lunak pengembangan

Software yang digunakan untuk pengembangan perangkat lunak sistem konversi file XML dan manipulasinya adalah sebagai berikut:

a. Microsoft Windows XP Professional 2002
b. Microsoft Visual Basic 6.0
c. Microsoft Word 2003
d. Microsoft Visio 2003
e. Microsoft Notepad
f. Adobe Photoshop CS2
g. ACDSee Foto Canvas versi 3.0.2.0007
h. HTML Help Workshop versi 4.74. 8702.0
i. Macromedia Dreamweaver MX
j. InstallShield 10.5

k. Microsoft Developer Network 2001

Berikut adalah perangkat lunak yang dibutuhkan untuk menjalankan aplikasi yang dihasilkan:

a. Sistem operasi Windows 98 dan sesudahnya

b. Microsoft XML Versi 4.0

5.1.2 Kebutuhan perangkat keras

Kebutuhan perangkat keras untuk pengembangan aplikasi dalam penelitian ini adalah sebagai berikut:

a. *Processor* Intel Pentium M 1500 MHz

b. Memori DDRAM 256 MB

c. Intel 82852 Internal Graphic Controller

d. *Monitor*, *Keyboard*, dan *mouse*

Spesifikasi minimal perangkat keras yang diperlukan untuk menjalankan aplikasi tersebut adalah sebagai berikut:

a. Processor pentium III 800 MHz atau yang setara

b. Memori minimal DDRAM 128 MB atau yang setara

c. Ruang Hardisk 40 MB

d. *Monitor*, *Keyboard*, dan *mouse*

5.2 Implementasi Perangkat Lunak

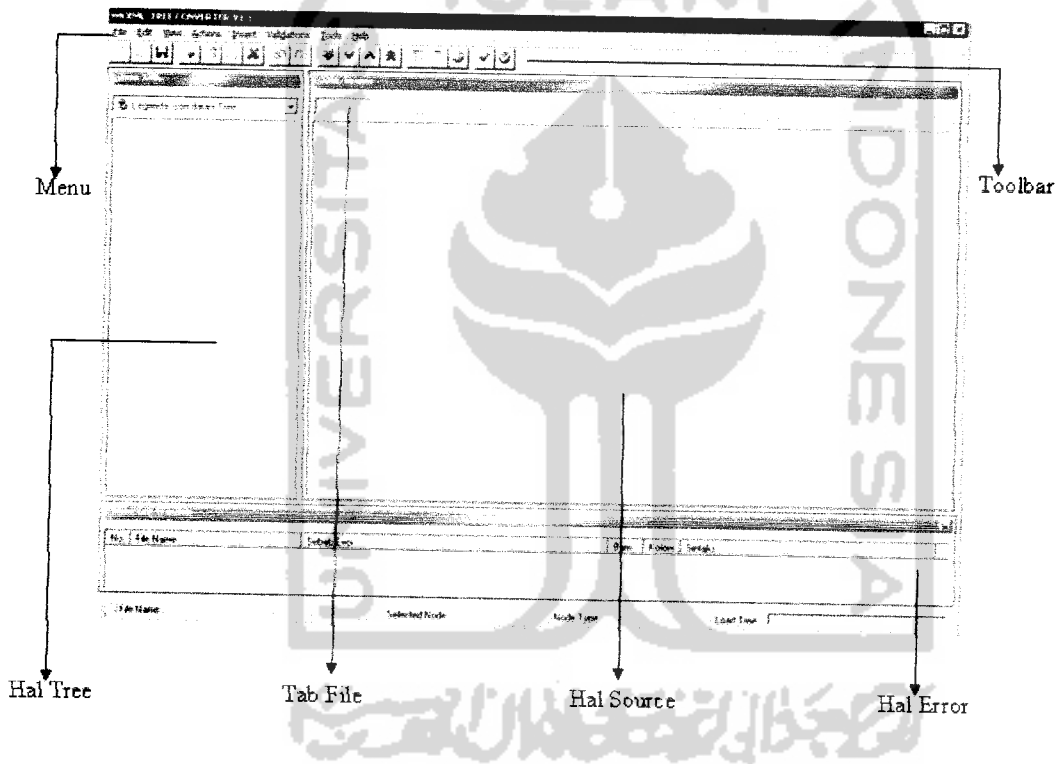
Bab ke lima ini akan menjelaskan mengenai implementasi perangkat lunak berdasarkan pada analisis kebutuhan dan perancangan yang sudah diterangkan dalam dua

bab terdahulu. Implementasi akan menggunakan perangkat lunak pengembangan dan bahasa Visual Basic 6.0.

5.3 Implementasi antarmuka

5.3.1 Tampilan Utama

Tampilan utama sistem konversi dan manipulasi XML diilustrasikan pada gambar 5.1 berikut ini:



Gambar 5.1 Tampilan utama

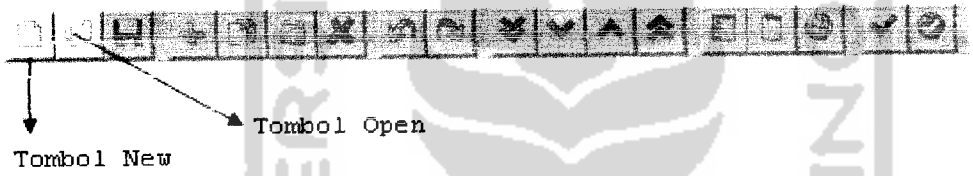
Menu dan toolbar menjadi alat navigasi utama dalam menjalankan aplikasi ini.

Halaman *tree* berfungsi untuk menampilkan visualisasi *tree* dari node-node di dalam file XML yang memenuhi syarat wellformed. Jika file yang terbuka belum memenuhi status wellformed atau berekstensi selain XML maka halaman *tree* ini akan dikosongkan.

Tab file berfungsi menampilkan nama file lengkap dengan ekstensinya. Tab file mempunyai icon yang berbeda untuk jenis file yang berbeda. Halaman source berisi source code dari file yang dibuka. Halaman error merupakan tempat untuk menampung semua keterangan kesalahan-kesalahan yang terjadi saat aplikasi digunakan.

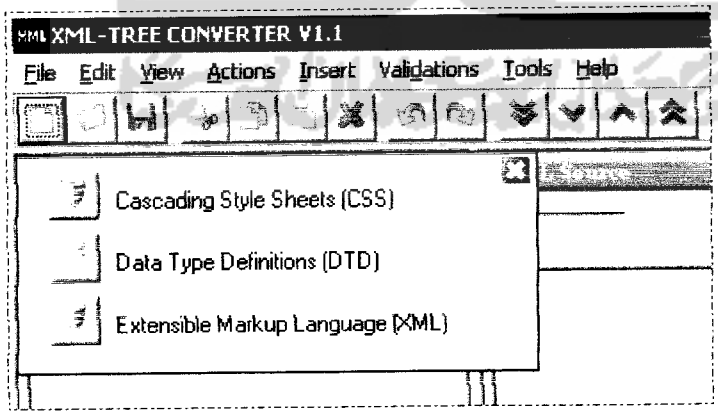
5.3.2 Membuka dan membuat file

Aplikasi mampu membuka dan membuat tiga jenis file yaitu CSS, DTD, dan XML. Fasilitas ini dapat digunakan melalui menu File| open untuk membuka atau menu File| New untuk membuat file baru. Fasilitas ini juga bisa digunakan melalui toolbar seperti tiperlihatkan dalam gambar 5.2.



Gambar 5.2 Tombol New dan Open pada toolbar

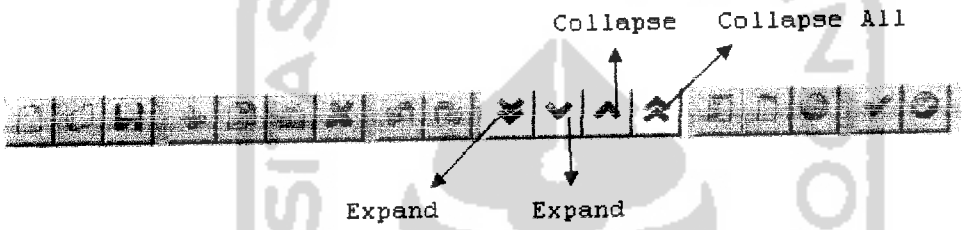
Pemilihan tombol new pada toolbar akan memicu terbukanya form new yang berfungsi untuk menawarkan pilihan jenis file yang akan dibuat seperti terlihat pada gambar 5.3.



Gambar 5.3 Form pilihan jenis file baru

5.3.4 Expand dan collapse tree

Fasilitas Expand node akan memudahkan user untuk melihat node-node yang posisinya di bawah suatu node. Disamping itu di dalam aplikasi juga terdapat fasilitas untuk menyembunyikan isi yang terkandung dari node, yaitu *Collapse* node. Fasilitas ini dapat diakses melalui menu File| Actions, atau melalui toolbar seperti terlihat pada gambar 5.5.

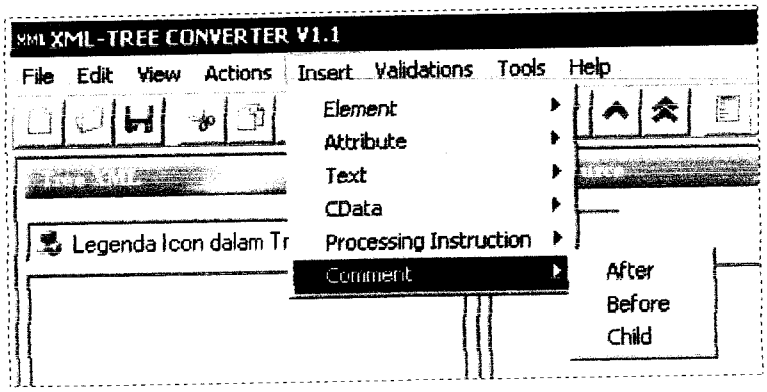


Gambar 5.5 Tombol expand dan collapse pada toolbar

Expand All merupakan fungsi untuk menampilkan seluruh node yang berada di dalam *tree*. Collapse All berfungsi untuk menyembunyikan seluruh node yang berada di bawah *root*.

5.3.5 Insert dan Edit Node

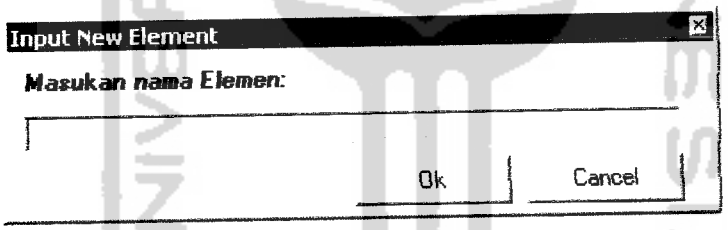
Penambahan node dalam *tree* dapat dilakukan melalui proses insert yaitu dengan melalui menu konteks, dengan cara meng-klik kanan mouse pada node di dalam *tree* atau dengan menekan menu Insert kemudian diikuti dengan memilih jenis node yang akan dibuat. Gambar 5.6 adalah menu insert dan jenis node yang tersedia dalam menu insert beserta cara penempatannya.



Gambar 5.6 Menu Insert dan jenis node yang tersedia dalam menu

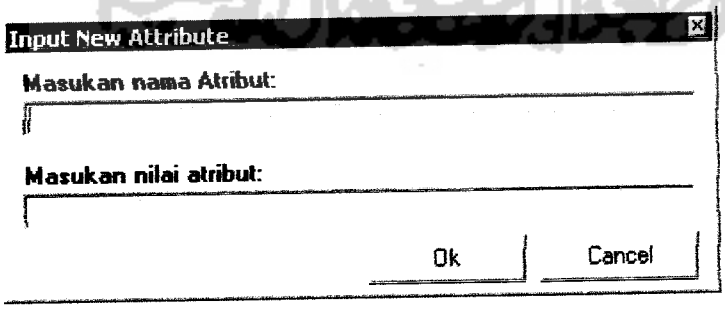
Kemudian ketika kursor berada di atas nama jenis node akan muncul pilihan penempatan node tersebut yaitu sejajar dengan node terpilih atau menjadi child.

Pemilihan jenis node ini akan memicu tampilnya form insert node. Form insert ini akan mempunyai kolom pengisian data untuk node baru. Node yang berbeda jenis dapat mempunyai kolom yang berbeda. Gambar 5.7 menunjukkan contoh form insert elemen.



Gambar 5.7 Form New Element

Gambar 5.8 menunjukkan form insert atribut.

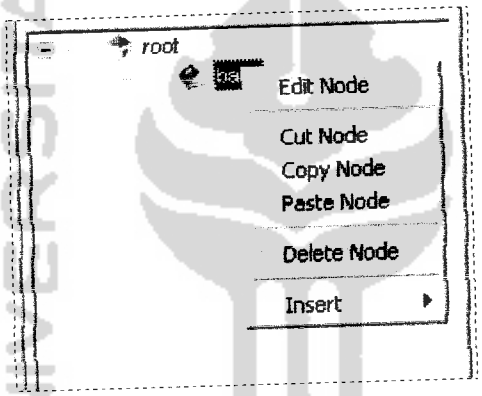


Gambar 5.8 Form New Atribut

Proses edit node dapat dilakukan dengan meng-klik kanan mouse pada node yang akan di-edit, kemudian dari menu konteks dipilih "edit node". Setelah node dipilih maka akan muncul form edit. Form ini hampir sama dengan form insert, hanya saja pada kolom pengisian sudah terisi data node yang lama. Setelah itu data yang lama dapat diubah.

5.3.6 Cut, Copy, Paste, dan Delete Node

Manipulasi cut, copy, paste, dan delete dapat dilakukan melalui menu aplikasi, toolbar, maupun menu konteks. Pilih node yang akan dimanipulasi, kemudian klik kanan, maka akan muncul menu konteks seperti diilustrasikan pada gambar 5.9.

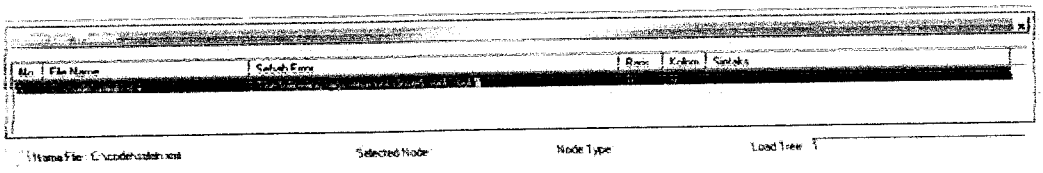


Gambar 5.9 Menu Konteks pada node dalam *tree*

Dari menu konteks tersebut dapat dipilih proses cut, copy, paste, atau menghapus suatu node.

5.3.7 Validasi XML

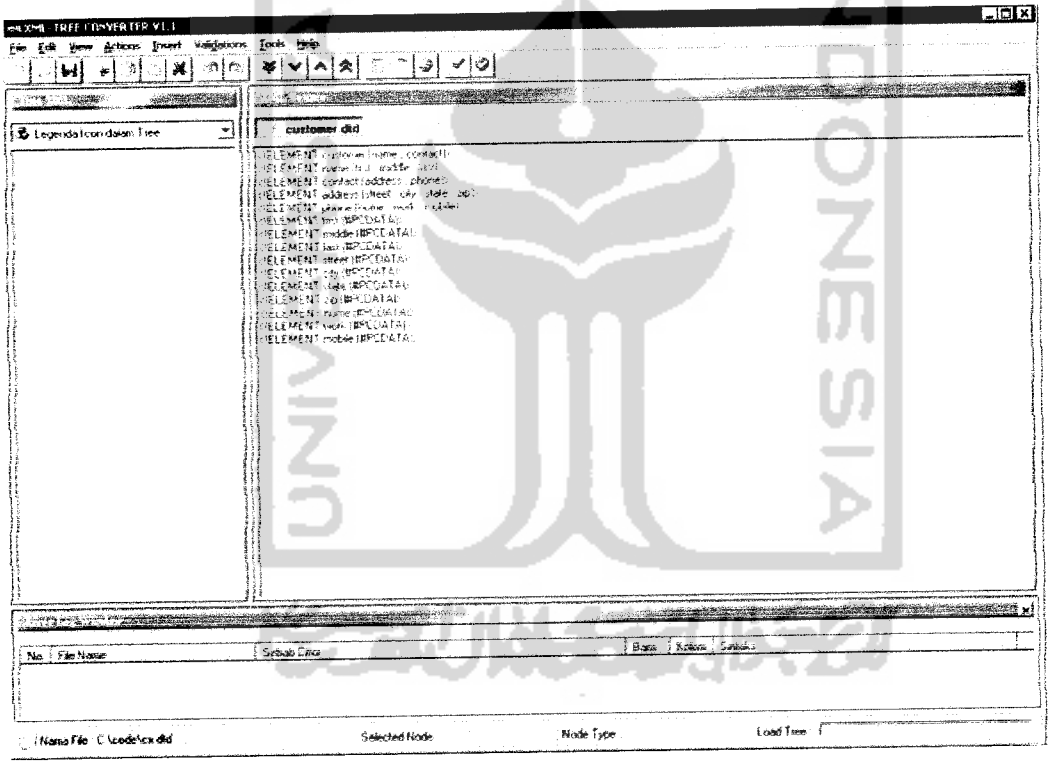
Validasi mode wellformed dan mode valid dapat dilakukan melaui toolbar ataupun melalui menu Validations. Jika file tersebut tidak valid maka akan muncul pesan kesalahan dan pada hal Error akan terdapat keterangan mengenai kesalahan yang terjadi. Gambar 5.10 menunjukkan halaman untuk menampung error yang pernah terjadi.



Gambar 5.10 Halaman penampung keterangan error

5.3.8 Tampilan untuk file DTD

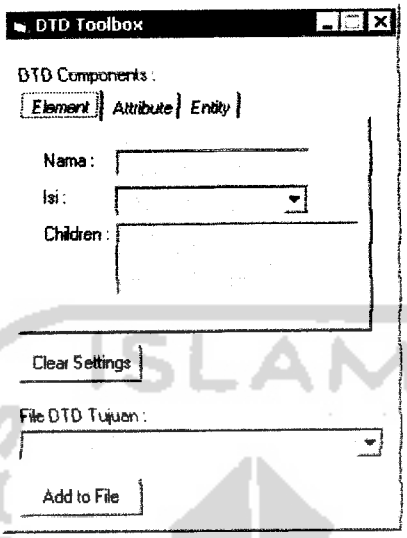
File DTD tidak dapat divisualisasikan dalam bentuk *tree*, sehingga ditampilkan oleh aplikasi dalam bentuk source code. Gambar 5.11 merupakan form utama yang menampilkan file DTD.



Gambar 5.11 Halaman utama untuk file DTD

Dalam gambar di atas dapat dilihat halaman *tree* tidak terisi, hal ini dikarenakan file DTD tidak dapat diubah oleh *parser* MSXML ke dalam struktur *tree*. MSXML hanya memakai file DTD sebagai validator XML.

Aplikasi menyediakan sebuah form untuk membantu user dalam membuat file DTD. Gambar 5.12 mengilustrasikan form tersebut.

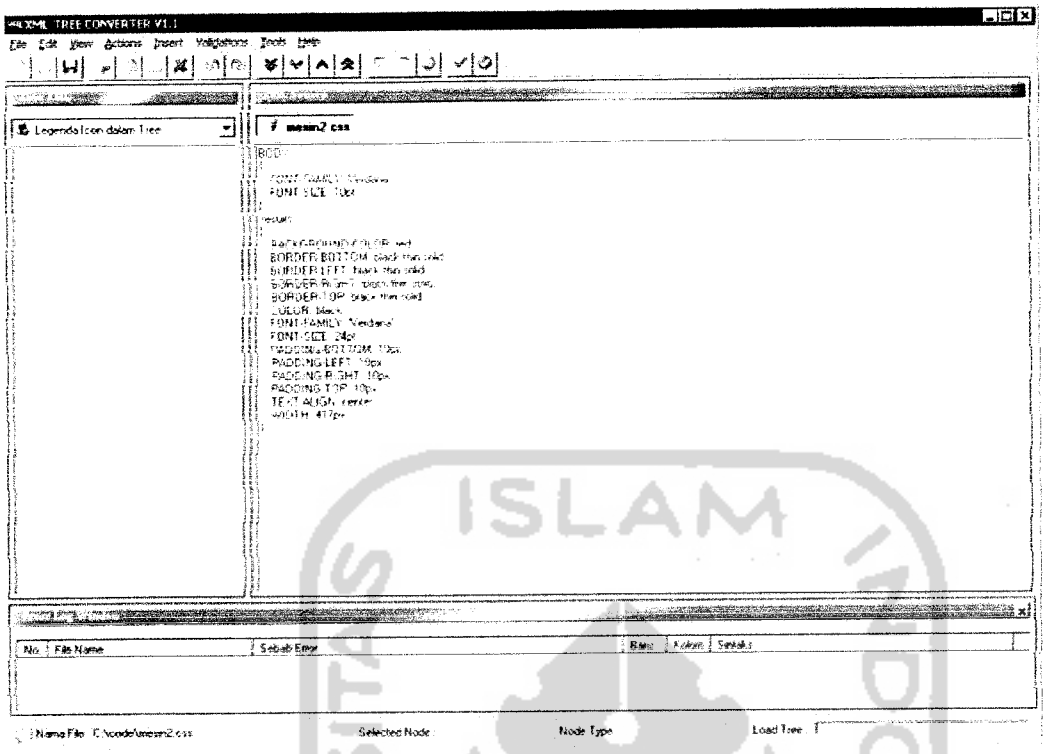


Gambar 5.12 Form pembuat DTD

Setelah user mengisi kolom-kolom komponen DTD dan memilih file DTD yang dituju, maka komponen DTD yang baru ditambahkan ke dalam file DTD yang dituju.

5.3.9 Tampilan untuk file CSS

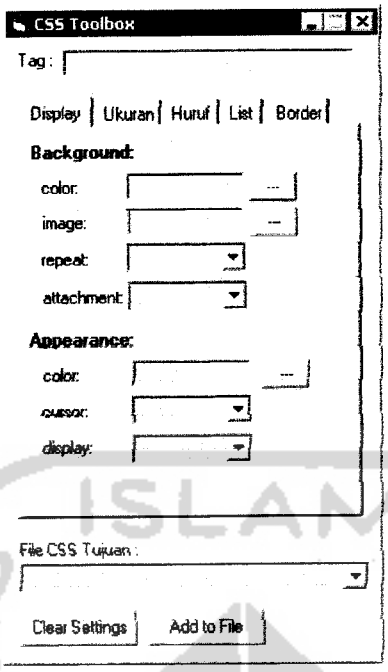
File CSS tidak dapat divisualisasikan dalam bentuk *tree*, sehingga ditampilkan oleh aplikasi dalam bentuk source code. Gambar 5.13 merupakan form utama yang menampilkan file CSS.



Gambar 5.13 Halaman utama untuk file CSS

Dalam gambar di atas dapat dilihat halaman *tree* tidak terisi, hal ini dikarenakan file CSS tidak dapat diubah oleh *parser* MSXML ke dalam struktur *tree*. MSXML hanya memakai file CSS sebagai media untuk memformat tampilan XML dalam *browser* internet

Aplikasi menyediakan sebuah form untuk membantu user dalam membuat file CSS. Gambar 5.14 mengilustrasikan form tersebut.

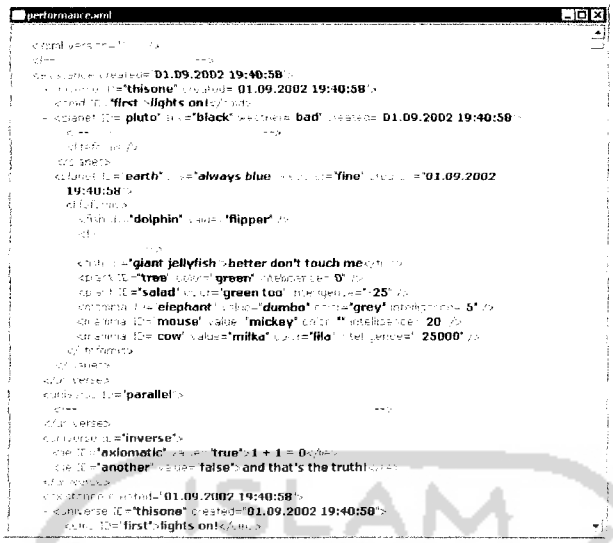


Gambar 5.14 Form pembuat CSS

Setelah user mengisi kolom *tag*, komponen CSS, dan memilih file CSS yang dituju, maka komponen CSS yang baru ditambahkan ke dalam file CSS yang dituju

5.3.10 Tampilan form Browser

File XML yang sudah tervalidasi dapat ditampilkan dalam *browser* internet. Untuk memudahkan user melihat dalam *browser* maka aplikasi dilengkapi dengan form *browser*. Gambar 5.15 merupakan form *browser* untuk file XML.



Gambar 5.15 Form browser internet

Browser ini mempunyai kegunaan hanya untuk menampilkan file XML yang sedang berjalan di dalam aplikasi XML-Tree converter.

5.4 Implementasi prosedur

Penulisan code program aplikasi dibagi menjadi beberapa prosedur, sehingga mempermudah dalam proses pembacaan dan penelusuran kesalahan yang terjadi. Berikut akan dijelaskan prosedur-prosedur utama dalam aplikasi yang dibangun.

5.4.1 Prosedur konversi XML ke dalam Tree

XML mempunyai node-node dengan ciri yang berbeda dan spesifik, node sebelum *root* akan berbeda sifat dengan yang terletak di bawah *root*, oleh karena diperlukan adanya prosedur penampil *tree* yang berbeda pula. Berikut merupakan prosedur yang berhubungan dengan konversi XML-Tree:

a. IsiTree

Prosedur IsiTree menerima input berupa file XML yang lulus uji validasi. Prosedur ini bertujuan menampilkan *root* dan node-node diatasnya dalam *tree*. Di dalam prosedur ini setiap node akan diberi *key* / kunci yang berisi urutan, level dan jenis node. Format key tersebut adalah "`:level:*jenis node`", tanda bintang (`*`) digunakan untuk memberi tanda pemisah antara level dengan jenis node. Sebagai contoh untuk *key* node pertama dalam *tree*: "`:0:*7`", artinya urutan node ke-1 (dalam *tree* nomor node dimulai dari 0) dan berjenis node processing instructions (untuk urutan jenis node lihat Tabel 4.1).

Kunci inilah yang nantinya akan menjadi acuan untuk proses identifikasi dan pemanggilan suatu node. Selanjutnya node tersebut akan diberi warna melalui prosedur *AmbilWarna* dan simbol melalui prosedur *AmbilGambar*. *Root* XML berjenis node elemen sehingga dapat mempunyai atribut serta child / anak. Ketika prosedur memproses *root*, maka prosedur akan memeriksa ada tidaknya node atribut, apabila ditemukan maka prosedur akan diberi kunci atau key agar dapat diidentifikasi. *Root* yang mempunyai child akan diproses lebih lanjut dalam prosedur *IsiTreeBawah*.

b. IsiTreeBawah

Prosedur ini menerima masukan berupa node yang mempunyai child. Prosedur *Tree* akan mengirimkan *root* yang mempunyai child ke dalam prosedur *IsiTreeBawah*. Selanjutnya *root* akan dijadikan sebagai acuan untuk memproses node child-nya, sehingga *root* akan dianggap sebagai *parent*. Seperti pada node di atas *root*, node-node di bawah *root* juga akan diberi *key*.

Node diberi *key* yang mengacu pada node *parent*-nya. Berikut contoh pemberian key dari child, misal key *parent* adalah "`:1:*1`", kemudian diambil levelnya saja melalui prosedur *GetLevelFromNodeKey* sehingga didapat "`:1:`". Pembuatan key untuk child

pertama yang berjenis elemen adalah sebagai berikut: level *parent* "`:1:`" ditambah level node tersebut ditambah dengan urutan jenis "`*1`" sehingga menjadi "`:1:0:*1`" Untuk child kedua yang berjenis komentar akan mempunyai kunci "`:1:1:*8`".

Pengecekan atribut akan kembali dilakukan untuk node yang berjenis elemen kemudian node tersebut diberi warna dan simbol sesuai jenisnya.Node akan dicek lagi untuk mengetahui ada tidaknya child di dalam node tersebut. Jika ditemukan child maka prosedur *IsiTreeBawah* akan diulang dengan node tersebut sebagai input.

c. IsiText

Prosedur ini menerima masukan berupa node yang mempunyai nilai. Nilai dari suatu node tidak ikut dimasukan dalam *tree*, tetapi ditempatkan disebelah kanan *tree* dalam komponen *listview*, untuk memasukan nilai ini dibutuhkan prosedur *IsiText*. Pertama yang akan dilakukan oleh prosedur ini adalah melihat seluruh nodedalam *tree* yang dalam keadaan *visible* / terlihat / kondisi expand. Setelah diketahui adanya node dalam *tree*, maka prosedur akan mencari node yang mempunyai nilai. Setelah itu nilai akan ditempatkan dalam listview dan diberi kunci yang sama dengan node pemilik nilai tersebut.

d. GetLevelFromNodeKey

Prosedur ini menerima masukan berupa kunci dari node. Level dari node akan menjadi acuan bagi node child-nya. Untuk memperoleh level tersebut dapat diambil dari kunci node. Karena di dalam kunci node tidak hanya berisi level, tapi juga terdapat jenis node, maka pengambilan level ini memerlukan prosedur *GetLevelFromNodeKey*. Prosedur ini mempunyai fungsi *InStr* yang berfungsi mengambil jumlah string dari posisi awal kunci sampai sebelum tanda "*". Kemudian hasilnya akan digunakan dalam fungsi

left untuk mengambil level, contoh: kunci ":1:1:*1" dikenai fungsi Instr sehingga didapat 5 string, kemudian dikenai fungsi left, didapat ":1:1:".

e. AmbilGambar

Prosedur ini akan mendapat masukan berupa node. Jenis node yang berbeda akan diberi simbol yang berbeda. Setelah itu icon diambil dari komponen penampung gambar, kemudian ditempatkan persis disebelah kiri node di dalam *tree*.

f. SetWarnaNode

Prosedur ini akan menerima input berupa node. Setelah itu node diperiksa jenisnya, selanjutnya setiap node akan diberi warna sesuai jenisnya. Settingan untuk warna ini tersimpan dalam registry.

g. SaveUndo

Seluruh tampilan ini akan disimpan dalam suatu file sementara, untuk keperluan *recovery* melalui prosedur *SaveUndo*. Sehingga bila *tree* berubah dan terjadi kesalahan, maka *tree* dapat dikembalikan ke posisi semula. Prosedur ini menerima masukan berupa indeks dari file yang aktif. Kemudian dari indeks tersebut dapat diketahui tampilan *tree*, kemudian tampilan ini diubah ke dalam file XML dan disimpan dalam folder: App.Path & "\temp\dokxml" & CStr(indeks) & "\"& namaFile & CStr(i) & ".xml".

h. Source code

Berikut merupakan prosedur untuk mengisi node root dan node yang terletak di atas root:

```
'*********************************************************************
Sub IsiTree(indeks As Byte, Optional noUndo As Boolean)
    Dim nodeT As MSComctlLib.node
    Dim nodex As IXMLDOMNode
    Dim jmlChild As Byte
    Dim nodeTipe As Byte
    Dim i As Byte
    Dim level As Byte
    Dim AnodeT As MSComctlLib.node
    Dim ListAttr As IXMLDOMNamedNodeMap
```

```
    Dim Attr As IXMLDOMAttribute
    Dim jmlAttr As Byte
    Dim AttrLevel As Byte
    Dim Attrkey As String

    With frmUtama
    .tvwXML.Nodes.Clear
    .lvwXML.ListItems.Clear
    .TeksSource.Text = dokXML(indeks).xml
    jmlChild = dokXML(indeks).childNodes.length

    For i = 0 To jmlChild - 1
        level = CStr(i)
        Set nodex = dokXML(indeks).childNodes(i)
        nodeTipe = nodex.nodeType
        If nodeTipe <> 10 Then
          Set nodeT = .tvwXML.Nodes.Add(, , ":" & level & ":*" & _
                                        nodeTipe, nodex.nodeName)
        Else
          Set nodeT = .tvwXML.Nodes.Add(, , ":" & level & ":*" & _
                                        nodeTipe, "#doctype")
        End If
        nodeT.Tag = "atas"
        Call Ambilgambar(nodeT, nodex)
        Call SetWarnaNode(nodeT, nodex)
        level = level + 1
    Next i
    If nodeTipe = 1 Then
        nodeT.Tag = "root"
        Set ListAttr = nodex.Attributes
            AttrLevel = 0
            For Each Attr In ListAttr
                Attrkey = getNodeKey(nodeT.key)
                Attrkey = Attrkey & ":" & AttrLevel & ":*" & 2
                Set AnodeT = .tvwXML.Nodes.Add(nodeT, tvwChild, _
                                Attrkey, Attr.nodeName)
                AnodeT.Tag = Attr.nodeType
                Call Ambilgambar(AnodeT, Attr)
                Call SetWarnaNode(AnodeT, Attr)
                AttrLevel = AttrLevel + 1
            Next Attr
        Call IsiNode(nodeT, nodex)
    End If
    Set .tvwXML.SelectedItem = nodeT
    If noUndo = True Then
        Exit Sub
    Else
        SaveUndo (indeks)
    End If
    End With
End Sub
'-------------------------------------------------------------
```

Berikut merupakan prosedur untuk mengisi node yang terletak di bawah root:

```
'*********************************************************************
Private Sub IsiNode(nodeT As MSComctlLib.node, nodex As IXMLDOMElement)
    Dim cNodeT As MSComctlLib.node
    Dim AnodeT As MSComctlLib.node
    Dim cNodeX As IXMLDOMNode
    Dim ListAttr As IXMLDOMNamedNodeMap
    Dim Attr As IXMLDOMAttribute
    Dim jmlAttr As Byte
    Dim AttrLevel As Byte
    Dim Attrkey As String
    Dim key As String
    Dim i As Integer
    Dim jmlChild As Byte
    Dim level As Byte

    With frmUtama
    jmlChild = nodex.childNodes.length

    For i = 0 To jmlChild - 1
        level = CStr(i)
        Set cNodeX = nodex.childNodes(i)
        key = getNodeKey(nodeT.key)
        key = key & ":" & level & ":*" & cNodeX.nodeType

        Set cNodeT = .tvwXML.Nodes.Add(nodeT, tvwChild, key,
cNodeX.nodeName)

        cNodeT.Tag = "bawah"
        Call Ambilgambar(cNodeT, cNodeX)
        Call SetWarnaNode(cNodeT, cNodeX)
        Select Case cNodeX.nodeType
            Case 1
                If cNodeX.Attributes.length <> 0 Then
                    Set ListAttr = cNodeX.Attributes
                    AttrLevel = 0
                    For Each Attr In ListAttr
                        Attrkey = getNodeKey(cNodeT.key)
                    Attrkey = Attrkey & ":" & AttrLevel & ":*" & 2
                        Set AnodeT = .tvwXML.Nodes.Add(cNodeT,_
tvwChild, Attrkey, Attr.nodeName)
                        AnodeT.Tag = "bawah"
                        Call Ambilgambar(AnodeT, Attr)
                        Call SetWarnaNode(AnodeT, Attr)
                        AttrLevel = AttrLevel + 1
                    Next Attr
                End If
            Call IsiNode(cNodeT, cNodeX)
        End Select
    Next i
    End With
End Sub
'-------------------------------------------------------------
```

Beberapa node dalam XML mempunyai value atau isi, untuk itu diperlukan tempat menampilkan isi node tersebut. Aplikasi akan menampilkan isi tepat disebelah kanan tree. Berikut prosedur untuk menampilkan isi node dalam sebuah listview.

```
'*********************************************************************
Sub IsiListView()
    Dim itemX As ListItem
    Dim nodex As IXMLDOMNode
    Dim nodeT As MSComctlLib.node
    Dim key As String
    Dim teks As String
    Dim indeks As Byte
    Dim i As Integer
    Dim nodeTpertama As MSComctlLib.node

    With frmUtama
        indeks = frmUtama.TabSource.SelectedItem.Tag
        ReDim tmpNodeT(0)
        Set nodeT = panggilFirstNodeT()
        Call IsiTmpNodeT(nodeT)
        .lvwXML.ListItems.Clear
        .TextItem.Visible = False
        For i = LBound(tmpNodeT) + 1 To UBound(tmpNodeT)
            Set nodeT = tmpNodeT(i)
            Set nodex = PanggilNodeX(nodeT)
            key = CStr(nodeT.index) & "k"
            If nodex.nodeType = 1 Then
                teks = ""
            ElseIf nodex.nodeType = 10 Then
                teks = nodex.xml
            Else
                teks = nodex.Text
            End If
            Set itemX = .lvwXML.ListItems.Add(i, key, teks)
            itemX.Tag = nodeT.key
            itemX.ForeColor = nodeT.ForeColor
        Next
        ReDim tmpNodeT(0)
    End With
End Sub
'---------------------------------------------------
```

5.4.2 Prosedur expand dan collapse node

Node dalam *tree* dapat ditampilkan melalui prosedur expand dan dapat disembunyikan dari *tree* dengan prosedur collapse. Prosedur yang berkenaan dengna proses expand dan collapse adalah prosedur tvwXML_Expand, tvwXML_Collapse,

panggilExpand, ExpandNode, CollapseNode, ExpandAll, CollapseAll, dan ExpandAll2. Berikut penjelasan untuk masing-masing prosedur di atas:

a. TvwXML_Expand

Prosedur dijalankan ketika ada node dalam *tree* yang dikenai proses expand. Prosedur menerima masukan berupa node yang di-expand tersebut. Kemudian child dari node tersebut akan ditampilkan, begitu juga dengan nilai dari child node akan ditampilkan di dalam listview. Status expand node tersebut selanjutnya disimpan di dalam registry dengan nilai *true*.

b. TvwXML_Collapse

Prosedur dijalankan ketika ada node dalam *tree* yang dikenai proses collapse. Prosedur menerima masukan berupa node yang di-collapse tersebut. Kemudian child dari node tersebut akan disembunyikan, begitu juga dengan nilai node akan disembunyikan dari listview. Status expand node tersebut selanjutnya disimpan di dalam registry dengan nilai *false*.

c. PanggilExpand

Tab file berubah berarti file yang aktif-pun akan berubah, sehingga isi *tree*-pun akan berganti atau menjadi kosong. Kemudian ketika file yang pertama dipilih kembali, maka *tree* harus pada posisi seperti sebelumnya, oleh karena itu setiap kali node dikenai expand maupun collapse statusnya harus disimpan. Pemanggilan status node tersebut dapat dilakukan dengan prosedur PanggilExpand.

Prosedur ini dijalankan ketika *tree* sudah terlihat pada halaman *tree*. Kemudian prosedur ini mendapat masukan berupa indeks tab file, kemudian dari indeks tersebut

dipanggil status node dalam registry. Jika status suatu node adalah *true* maka node tersebut akan di-expand, namun jika *false* maka node tersebut akan collapse.

d. ExpandNode dan CollpaseNode

Prosedur ini membuat suatu node di-expand atau di-collapse.

e. ExpandAll, collapseAll, dan expandAll2

Prosedur ini membuat seluruh node expand atau collpase. Sedangkan node expandAll2 berfungsi untuk melakukan perulangan untuk memberi status node dari awal sampai akhir node.

f. Source Code

```
'*****************************************************************
Sub Expand(jmlnode As Long, _
           ListNodeT As MSComctlLib.Nodes, Bool As Boolean)
    Dim nodeT As MSComctlLib.node
    Dim i As Long
    Dim ibar As Integer

    On Error Resume Next

    With frmUtama.ProgressBar1
        ibar = 1000 / jmlnode
        For i = 1 To jmlnode
            Set nodeT = ListNodeT(i)
            If Bool = True Then
                nodeT.Expanded = True
            Else
                nodeT.Expanded = False
            End If
            .Value = .Value + ibar
        Next i
        .Value = 1000
        .Value = 1.0002
    End With
End Sub
'-----------------------------------------------------------------

'*****************************************************************
Private Sub tvwXML_Collapse(ByVal node As MSComctlLib.node)
    Dim indeks As String

    indeks = TabSource.SelectedItem.Tag
    SaveSetting App.EXEName, indeks, node.key, False
    IsiListView
End Sub
```

```
Private Sub tvwXML_Expand(ByVal node As MSComctlLib.node)
    Dim indeks As String

    indeks = TabSource.SelectedItem.Tag
    StatExpand(indeks) = True
    SaveSetting App.EXEName, indeks, node.key, True
    IsiListView
End Sub
'-----------------------------------------------------------------
```

Sedangkan berikut ini adalah prosedur untuk memanggil status expand / collapse node yang telah disimpan:

```
'*****************************************************************
Sub PanggilExpand(indeks)
    Dim VarSet As Variant
    Dim ExIndeks As String
    Dim NodeKey As String
    Dim ExValue As Boolean
    Dim i As Integer
    Dim ibar As Integer

    On Error Resume Next

    ExIndeks = CStr(indeks)
    VarSet = GetAllSettings(App.EXEName, ExIndeks)
    NodeKey = VarSet(0, 0)
    ExValue = VarSet(0, 1)
    ibar = 0

    With frmUtama
    .ProgressBar1.Value = frmUtama.ProgressBar1.Value + ibar
    For i = 0 To UBound(VarSet, 1)
        NodeKey = VarSet(i, 0)
        ExValue = VarSet(i, 1)
        If .tvwXML.Nodes.Item(NodeKey).Expanded <> ExValue Then
            ibar = ibar + 1
        End If
    Next i
    ibar = 1000 / ibar
    For i = 0 To UBound(VarSet, 1)
        NodeKey = VarSet(i, 0)
        ExValue = VarSet(i, 1)
        If .tvwXML.Nodes.Item(NodeKey).Expanded <> ExValue Then
            .tvwXML.Nodes.Item(NodeKey).Expanded = ExValue
        End If
        .ProgressBar1.Value = frmUtama.ProgressBar1.Value + ibar
    Next i
    .ProgressBar1.Value = 1
    End With
End Sub
'-----------------------------------------------------------------
```

5.4.3 Prosedur untuk cut, copy, paste, dan delete node

Node di dalam *tree* selanjutnya dapat dimanipulasi agar didapat hasil yang sesuai dengan kebutuhan user. Manipulasi node yang tersedia dalam aplikasi ini meliputi proses cut, copy, paste, dan delete. Proses cut dengan copy mempunyai alur yang hampir sama, sehingga proses cut dan copy mengacu pada prosedur yang sama.

a. Copy

Prosedur copy menggunakan sebuah variabel global untuk menyimpan nilai node yang akan dicopy. Setelah sebuah node ditunjuk dan dicopy maka nilainya akan masuk ke dalam variabel global. Nilai varibel tersebut akan tetap disimpan selama program masih berjalan.

b. CopyChildT dan CopyChildX

Jika node yang dikenai proses copy mempunyai child, maka prosedur CopyChildT dan CopyChildX akan dijalankan. Prosedur ini membaca node yang dicopy, kemudian melakukan perulangan untuk membaca seluruh child dan menyimpanya ke dalam varibel global.

c. Cut

Proses cut hampir tidak mempunyai perbedaan dengan proses copy. Penggunaan variabel global untuk menyimpan data node yang dicut dan perulangan untuk membaca child dan menyimpanya. Perbedaannya adalah pada node yang dikenai copy akan tetap ada, sedangkan pada node yang dikenai cut akan dihapus dari *tree*.

d. Paste

Setelah node dikenai proses cut atau copy maka node tersebut dapat dipaste-kan menjadi child atau *sibling* node lainnya. Namun dalam proses paste ini diharus melewati beberapa persyaratan pembuatan node, misal node processing instruction dan node doctype tidak boleh berada di atas *root*, node atribut tidak mempunyai anak, dsb.

e. Delete

Prosedur ini berfungsi untuk menampilkan pesan peringatan pada user bahwa proses penghapusan akan dijalankan. Kemudian user diberi pilihan untuk melanjutkan atau membatalkan perintah penghapusan. Jika dilanjutkan maka node tersebut akan dikirimkan ke prosedur DeleteNodeX.

f. DeleteNodeX

Prosedur ini menerima masukan node dari prosedur delete. Kemudian prosedur ini akan menghapus node tersebut dari *tree*. Node di dalam *tree* akan menjadi berkurang sehingga perlu dilakukan penyusunan kunci dari node-node didalamnya. Setelah itu isi dari ListView juga disesuaikan dengan isi *tree* yang baru.

5.4.4 Prosedur untuk validasi

Prosedur validasi bertujuan untuk untuk mengetahui kebenaran file XML. Prosedur validasi terbagi menjadi prosedur validasi wellformed dan validasi valid. Wellformed berkaitan dengan kebenaran sintaks sedang validasi valid berkaitan dengan kebenaran isi. Berikut prosedur untuk validasi wellformed:

a. Wellformed

Prosedur ini menerima masukan berupa file XML, kemudian memerintahkan *parser* untuk melakukan validasi wellformed. Jika file sudah divalidasi selanjutnya prosedur ini akan menampilkan pesan pada user apakah file tersebut wellformed atau

tidak. Jika tidak wellformed maka file tidak dapat ditampilkan di dalam *tree*, sehingga akan ditampilkan dalam bentuk source code-nya saja.

b. Valid

Prosedur ini hampir sama dengan prosedur wellformed, perbedaanya terletak pada perintah yang diberikan pada *parser*. Prosedur ini memerintahkan *parser* untuk melakukan validasi valid.

BAB VI

ANALISIS KINERJA

Analisis kinerja perangkat lunak digunakan untuk menguji validitas perangkat lunak yang dihasilkan, terutama fungsi-fungsi yang ada didalamnya. Setelah melewati proses analisis ini, diharapkan perangkat lunak yang dihasilkan dapat memenuhi kebutuhan yang diharapkan.

Pengujian GUI atau *Graphical User Interface* bertujuan memeriksa kebenaran dari antarmuka yang dihasilkan. Kebenaran yang dimaksud adalah kesesuaian antara antarmuka terhadap fungsi-fungsi yang seharusnya ada dan diakomodasi oleh antarmuka tersebut. Antarmuka yang dimaksud meliputi *windows, icon, menu,* dan *pointers.*

6.1 Pengujian proses konversi XML-Tree

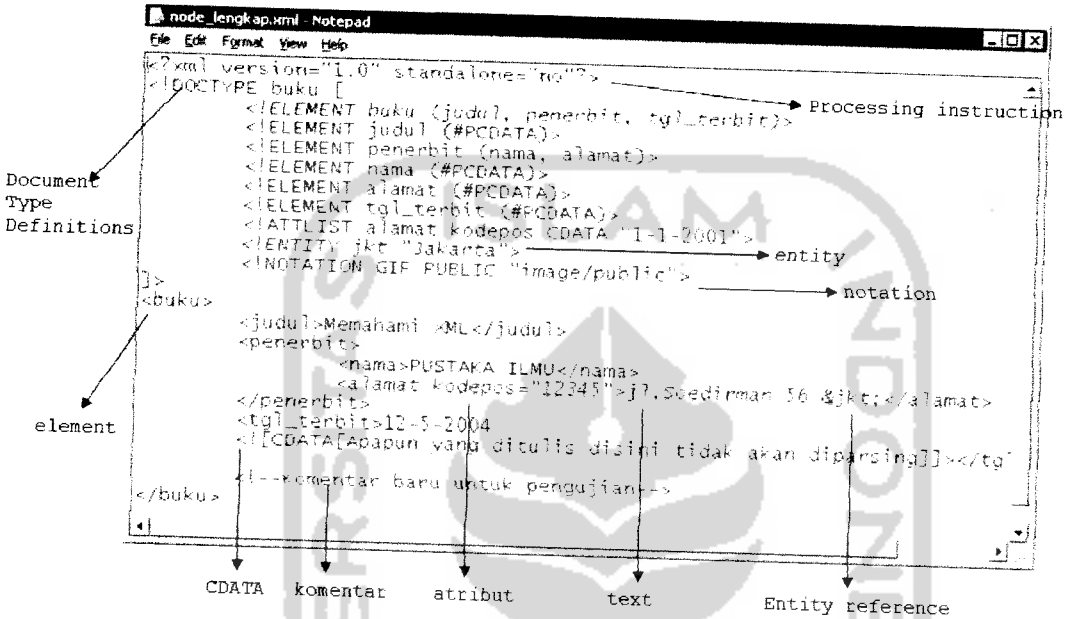
Proses konversi XML-Tree merupakan proses utama sebelum proses yang lain dapat dilakukan. Proses ini memerlukan *parser* MSXML 4.0 untuk mengubah file XML menjadi *tree* secara logika. Aplikasi kemudian melakukan visualisasi file tersebut ke dalam tampilan *tree* sehingga dapat dilihat dan dimanipulasi oleh *user.*

6.1.1 Konversi file XML yang sudah ada

Aplikasi memanggil file XML yang telah dibuat dan disimpan dalam suatu folder dengan menu File| Open atau melalui toolbar. Setelah nama file ditentukan maka file akan di-validasi terlebih dahulu agar diketahui statusnya.

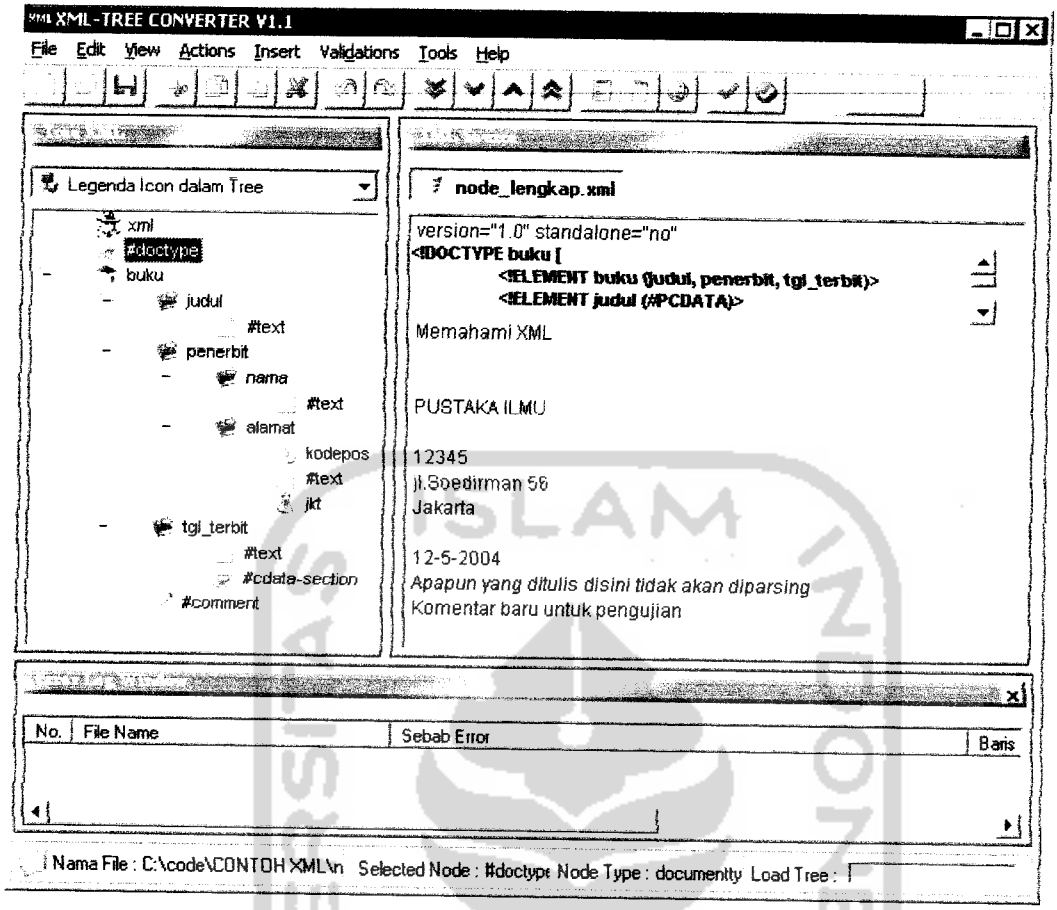
a. Pengujian dengan data normal

Parser MSXML membedakan node file XML menjadi 12 jenis seperti yang telah dijelaskan di dalam BAB IV sub 4.3. *Parser* MSXML 4.0 mampu membaca semua jenis node tersebut, namun tidak seluruh jenis node tersebut dapat ditampilkan dalam *tree*. Gambar 6.1 menunjukan sebuah contoh file XML dengan 10 jenis node didalamnya.



Gambar 6.1 File "node_lengkap" dalam notepad dengan 10 jenis node

Node yang tidak ikut diuji adalah node document dan node fragment. Node document tidak digunakan karena node tersebut sebenarnya adalah file XML itu sendiri, sehingga yang dipakai oleh aplikasi adalah isi dari node document. Sedangkan node fragment adalah fungsinya sama dengan node entity yang dipanggil oleh node entity reference. Gambar 6.2 menunjukan struktur *tree* dari file node_lengkap.xml yang dihasilkan oleh aplikasi.

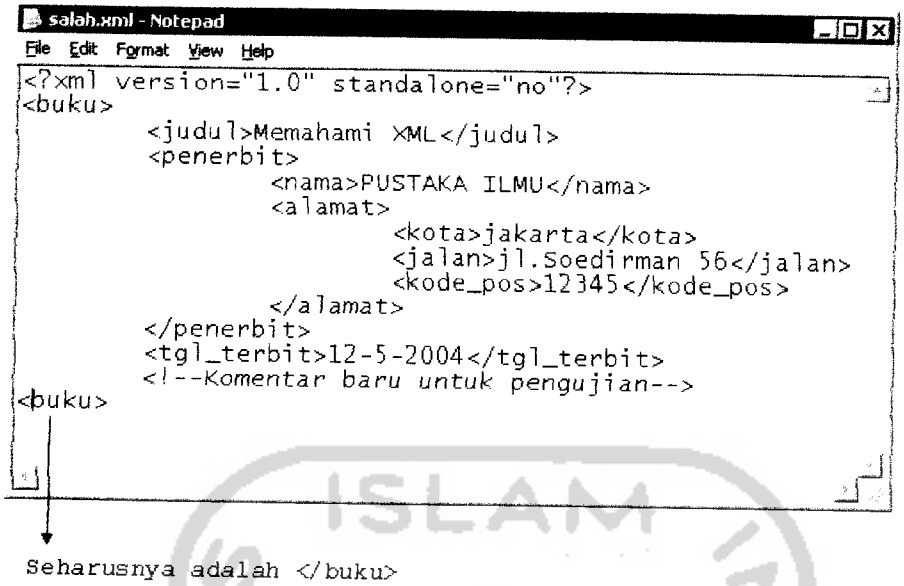


Gambar 6.2 Struktur *tree* dalam aplikasi

Nilai dari sebuah node terkadang sangat panjang sehingga kurang tepat bila ikut dimasukan ke dalam *tree*. Nilai tersebut ditempatkan disebelah kanan dari node yang memilikinya dalam sebuah komponen ListView.

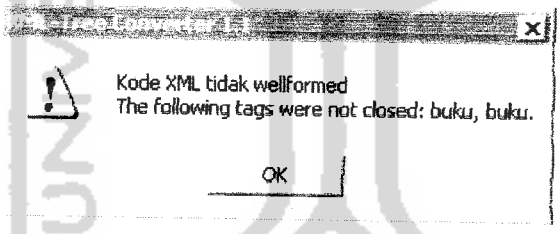
b. Pengujian dengan data tidak normal

Apabila file tersebut sesuai aturan sintaks maka akan langsung dikonversi ke dalam *tree*. Namun bila tidak sesuai dengan aturan sintaks akan muncul peringatan kesalahan. Pengujian dilakukan dengan sebuah file XML yang bernama "salah.xml" seperti terlihat pada gambar 6.3.



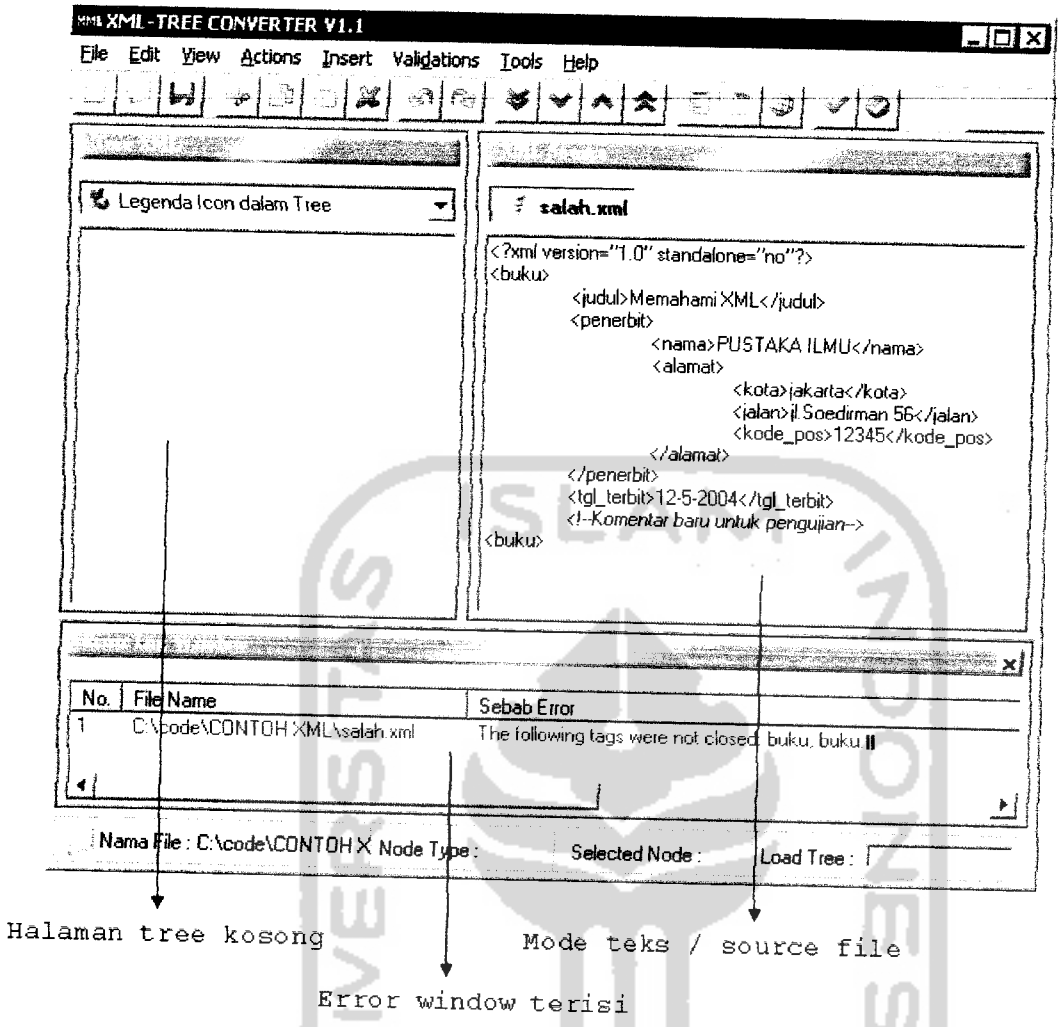
Gambar 6.3 File "salah.xml" dalam notepad

Pada gambar 6.3 dapat dilihat element <buku> ditutup oleh *tag* <buku>, seharusnya ditutup dengan *tag* </buku>, sehingga akan terjadi kesalahan yang menimbulkan pesan error seperti gambar 6.4.



Gambar 6.4 Pesan file XML tidak wellformed

Pesan peringatan pada gambar 6.4 juga berisi keterangan tentang kesalahan yang terjadi. Selanjutnya pesan kesalahan juga akan tercatat pada error log window seperti terlihat pada gambar. Kesalahan tersebut menyebabkan file tidak dapat dikonversi ke dalam *tree*, sehingga file akan ditampilkan dalam bentuk source code-nya dalam mode teks seperti terlihat pada gambar 6.5.



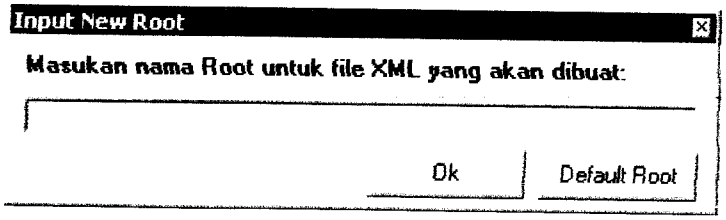
Gambar 6.5 Mode teks dan error window file "salah.xml"

6.1.2 Konversi file XML baru

Aplikasi mempunyai fasilitas untuk membuat file baru. Pembuatan file XML memicu tampilnya form untuk memasukan *root*, yaitu node yang merupakan salah satu syarat agar file XML menjadi wellformed dan valid.

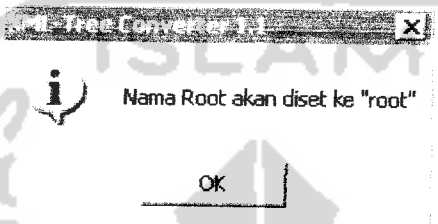
a. Pengujian dengan data normal

Pengujian pembuatan file XML baru dimulai dengan membuat *root* secara default, yaitu dengan menekan tombol "Default Root" seperti pada gambar 6.6.



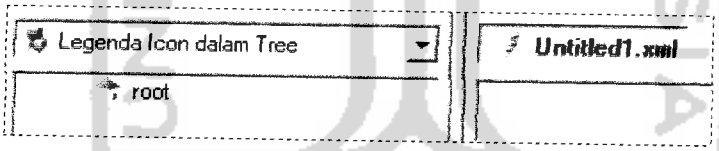
Gambar 6.6 Form input Root

Penekanan tombol "Default Root" akan memicu tampilnya pesan pembuatan *root*, seperti diperlihatkan pada gambar 6.7.



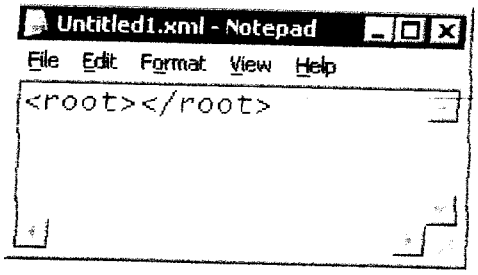
Gambar 6.7 Pesan informasi nama *root*

Selanjutnya setelah penekanan tombol "ok" maka secara otomatis di halaman *tree* akan muncul sebuah *tree* baru dengan sebuah node bernama "root". Gambar 6.8 menunjukan *tree* tersebut.



Gambar 6.8 *Tree* dari file XML baru

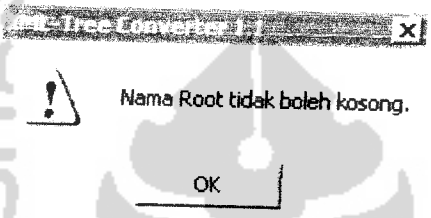
Secara default file tersebut akan diberi nama dengan awalan "Untitled" dan diakhiri dengan nomer urut pembuatan. Kemudian setelah itu disimpan dengan menekan tombol save, maka akan didapat file seperti gambar 6.9.



Gambar 6.9 File XML baru dalam notepad

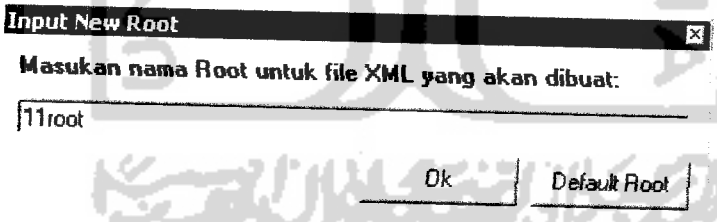
b. Pengujian dengan data tidak normal

Pada saat form input root muncul kemudian dilakukan penekanan tombol "OK" tanpa mengisi nama *root* akan memicu pesan peringatan seperti pada gambar 6.10.



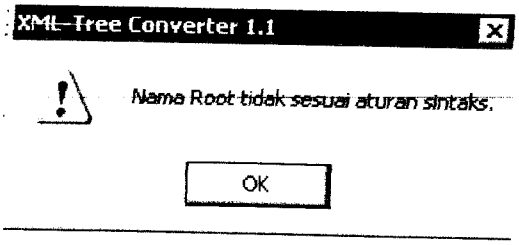
Gambar 6.10 Pesan kesalahan karena tidak mengisi nama *root*

Pengisian nama *root* yang tidak sesuai dengan aturan penamaan *root* juga akan menimbulkan pesan error. Seperti terlihat pada gambar 6.11.



Gambar 6.11 Nama *root* tidak sesuai aturan

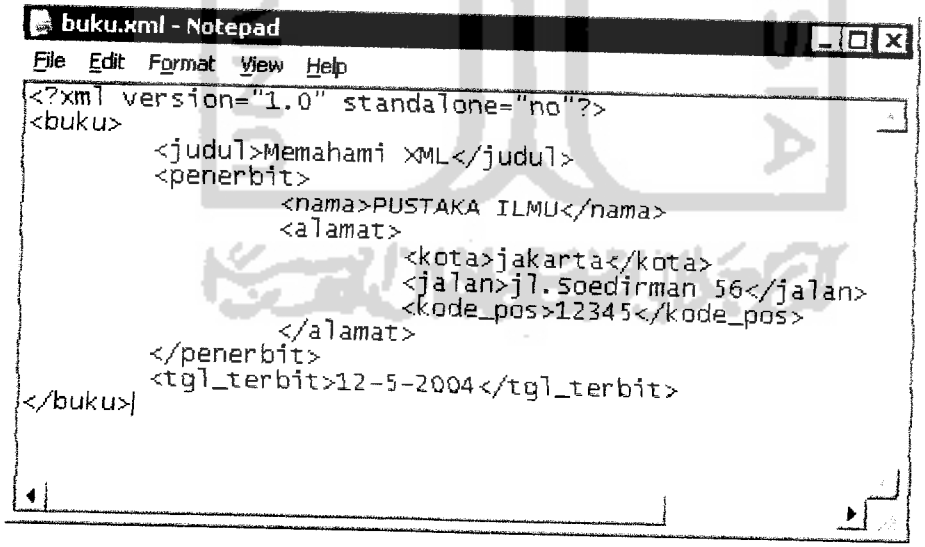
Pada gambar 6.11 nama *root* diawali dengan angka, maka ini dianggap sebagai kesalahan oleh aplikasi (untuk aturan penamaan node lihat BAB II sub 2.6). Sehingga akan muncul pesan seperti gambar 6.12.



Gambar 6.12 Pesan kesalahan nama *root*

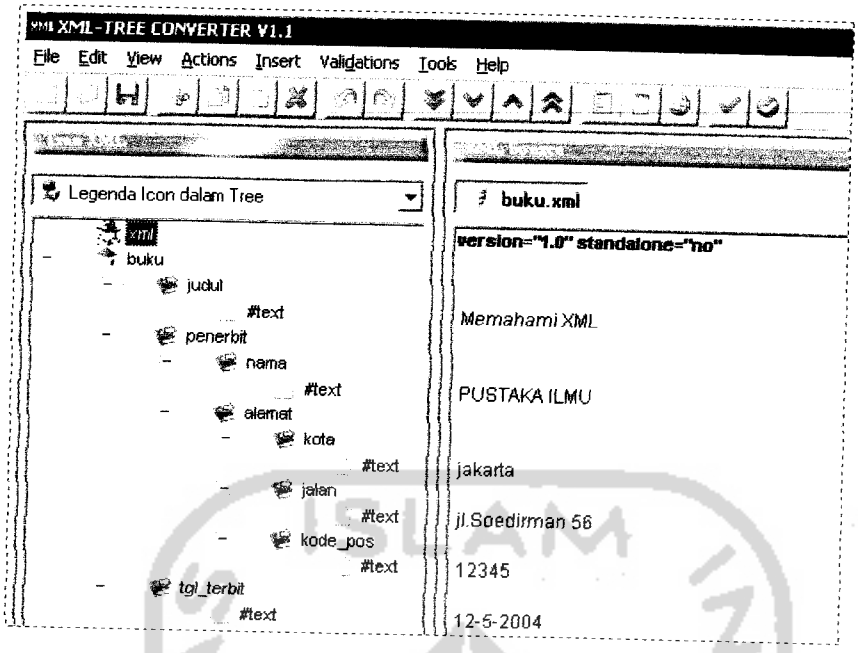
6.2 Pengujian proses insert

Pengujian ini menggunakan sebuah file XML dengan nama "buku.xml". Aplikasi belum mampu untuk melakukan proses insert dan edit node doctype, sehingga node yang berada dalam node doctype juga tidak dapat di-insert dan di-edit oleh aplikasi. Di dalam doctype terdiri dari file DTD internal, node entity, dan node notation. Oleh karena itu untuk pengujian selanjutnya digunakan file "buku.xml" yang mempunyai 7 jenis node yang dapat ditampilkan dalam *tree* sehingga dapat dikenai proses manipulasi. Seperti terlihat pada gambar 6.13 memperlihatkan file "buku.xml".



Gambar 6.13 File "buku.xml"

Setelah dikonversi ke dalam *tree* maka akan terlihat seperti pada gambar 6.14.

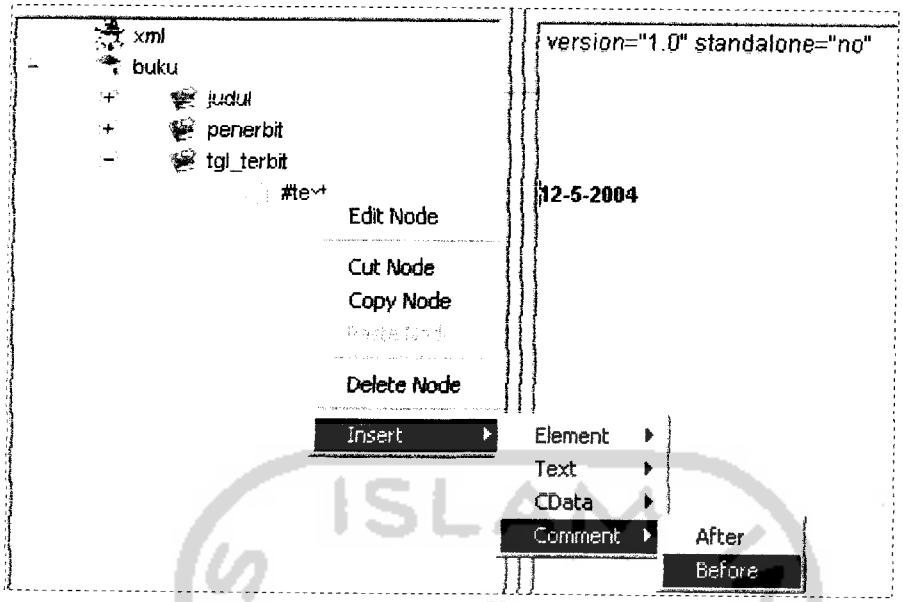


Gambar 6.14 Tree dari file "buku.xml"

a. Pengujian dengan data normal

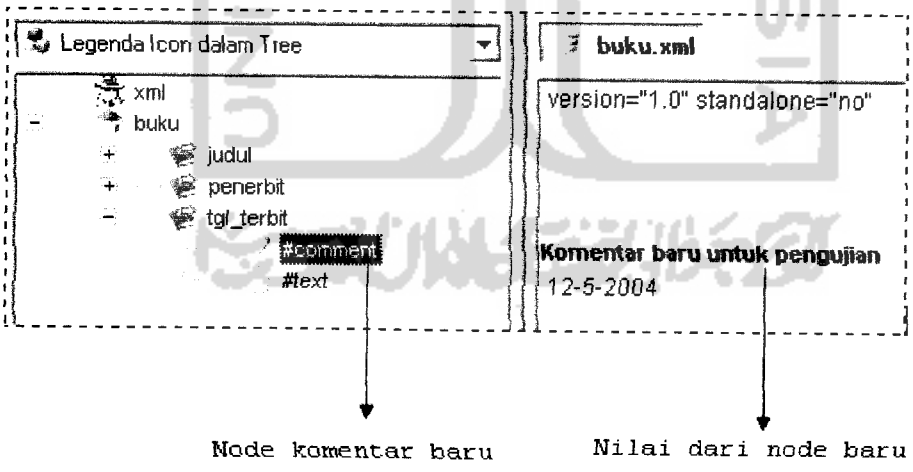
Node baru dapat disisipkan ke dalam *tree* melalui proses insert. Terdapat aturan mengenai penambahan node, untuk lebih jelasnya lihat di BAB 4 sub 4.3 tentang desain data. Aplikasi mempunyai kemampuan untuk menyeleksi node yang akan dimasukan ke dalam *tree*, sehingga membantu user agar tidak menempatkan node pada posisi yang salah.

Pengujian proses insert ini dilakukan dengan menggunakan menu konteks pada *tree* untuk membuat node comment, seperti yang terlihat pada gambar 6.15.



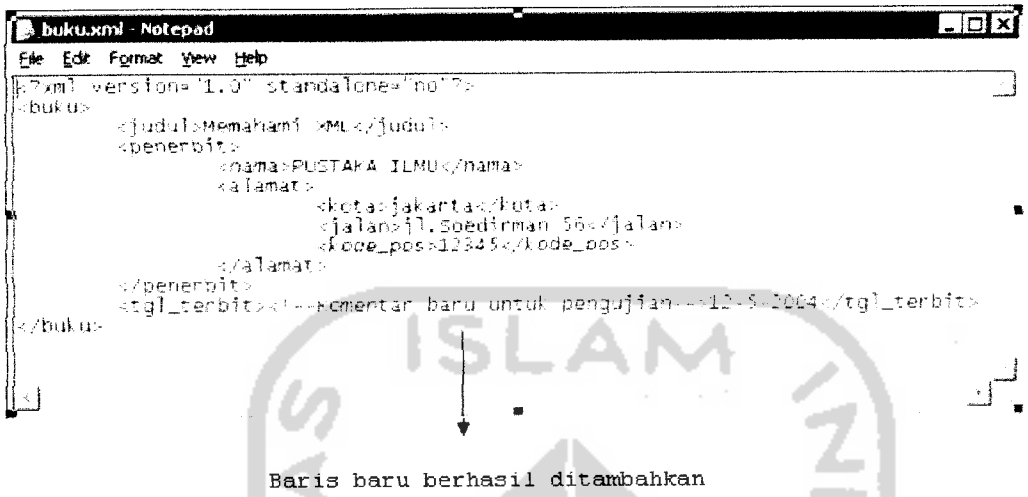
Gambar 6.15 Menu konteks insert pada *tree*

Pada gambar 6.16 terlihat menu konteks menunjuk pada Insert| Comment| before hal ini berarti sebuah node comment akan ditambahkan ke dalam *tree* diposisikan sebelum node #text yang merupakan anak dari node tgl_terbit. Form insert comment akan muncul dan setelah diisi maka node baru akan terbentuk, seperti terlihat pada gambar 6.16.



Gambar 6.16 Node comment disisipkan dalam *tree*

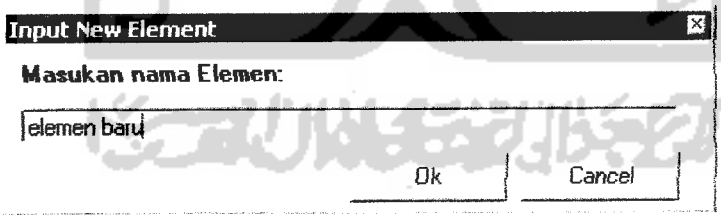
Kemudian setelah disimpan, file source XML akan berubah menjadi seperti pada gambar 6.17.



Gambar 6.17 Komentar berhasil disisipkan dalam file XML

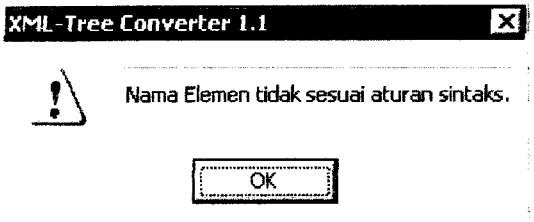
b. Pengujian dengan data tidak normal

Pengujian tidak normal ini dilakukan dengan memasukan nama node yang tidak sesuai aturan sintaks (untuk aturan penamaan node lihat BAB II sub 2.6). Gambar 6.18 menunjukan pengujian pembuatan node dengan nama "elemen baru", nama ini mengandung spasi didalamnya.



Gambar 6.18 Membuat node "elemen node"

Karena nama node tidak boleh mengandung spasi, maka akan muncul pesan kesalahan seperti pada gambar 6.19.



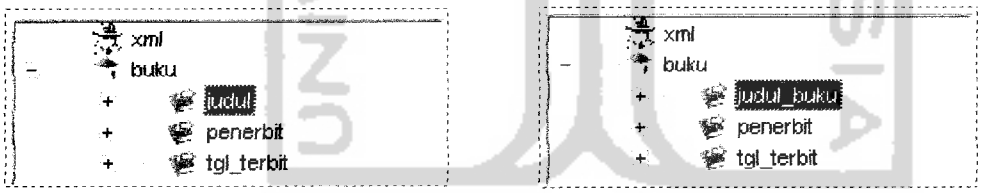
Gambar 6.19 Pesan kesalahan nama elemen

6.3 Pengujian proses edit

Proses edit bertujuan untuk mengubah nama node atau isinya. Proses ini hampir dapat dilakukan disemua node kecuali node doctype. MSXML 4.0 belum mempunyai kemampuan untuk mengubah node tipe doctype ini sehingga aplikasi yang dibuat juga belum mampu untuk meng-editnya.

a. Pengujian dengan data normal

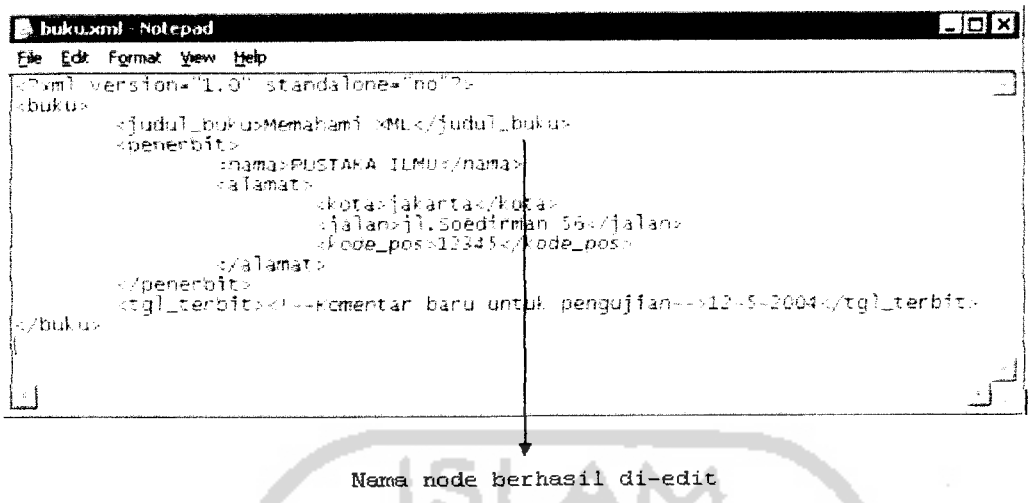
Pengujian dilakukan mengubah node judul menjadi judul_buku seperti terlihat pada gambar 6.20.



(node judul sebelum di-edit) (node judul setelah di-edit)

Gambar 6.20 Proses edit pada node judul

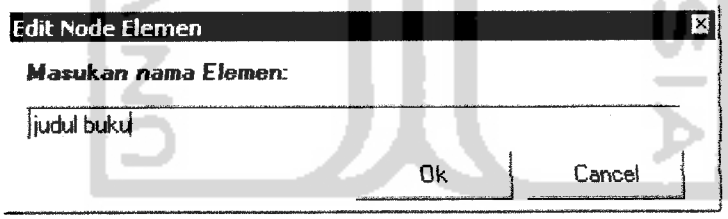
Setelah node berhasil dikenai proses edit dan kemudian disimpan, maka file XML akan berubah menjadi seperti gambar 6.21.



Gambar 6.21 Nama node "judul" berhasil diubah menjadi "judul_buku"

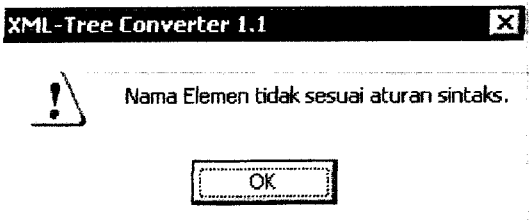
b. Pengujian dengan data tidak normal

Pengujian tidak normal ini dilakukan dengan memasukan nama node yang tidak sesuai aturan sintaks (untuk aturan penamaan node lihat BAB II sub 2.6). Gambar 6.22 menunjukan pengujian pembuatan node dengan nama "judul buku", nama ini mengandung spasi didalamnya.



Gambar 6.22 Edit node "judul" menjadi "judul buku"

Karena nama node tidak boleh mengandung spasi, maka akan muncul pesan kesalahan seperti pada gambar 6.23.



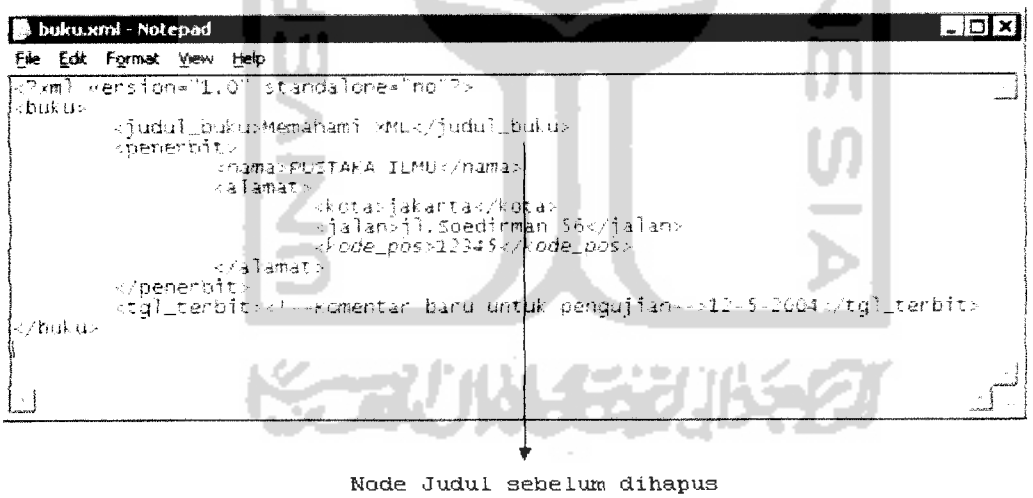
Gambar 6.23 Pesan kesalahan nama elemen

6.4 Pengujian proses delete

Node di dalam *tree* dapat juga dihapus yaitu dengan proses delete. Semua node dapat dihapus tanpa mengubah status validasi kecuali penghapusan node *root*.

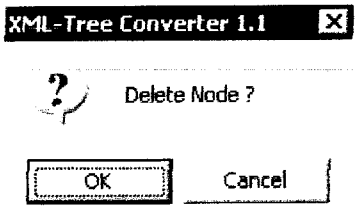
a. Pengujian normal

Pengujian dilakukan pada file buku.xml dengan menghapus node judul, seperti terlihat pada gambar 6.24.



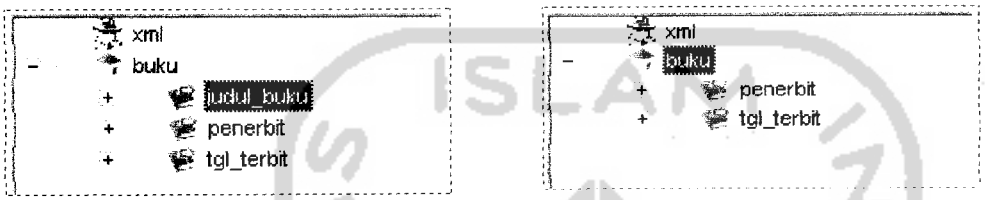
Gambar 6.24 Node judul pada file sebelum dihapus

Sebelum proses delete dijalankan, maka aplikasi akan menanyakan kepada user kesungguhan dalam menghapus node melalui pesan pada gambar 6.25.



Gambar 6.25 Pesan sebelum proses delete dilakukan

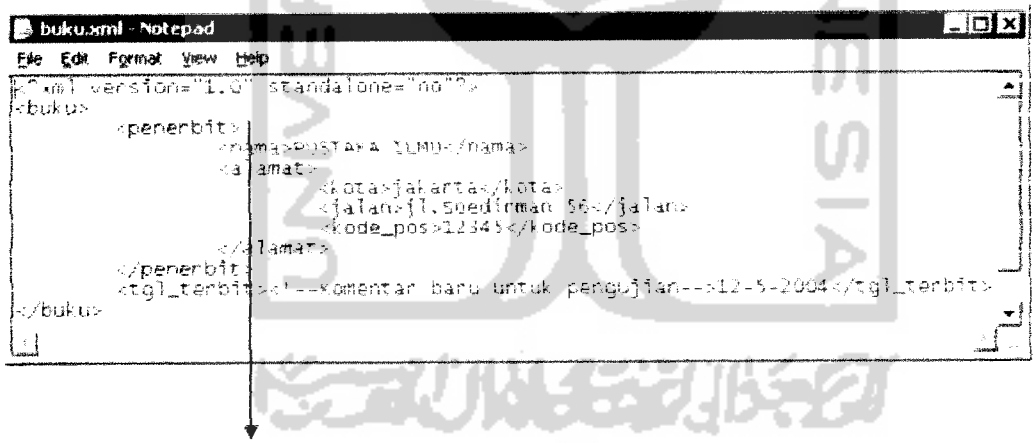
Pengujian proses delete node judul terlihat pada gambar 6.26.



(node judul sebelum di-delete) (node judul setelah di-delete)

Gambar 6.26 Node judul dikenai proses delete

Setelah disimpan maka baris judul pada file XML akan hilang, seperti pada gambar 6.27.

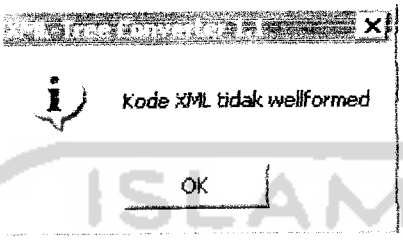


Gambar 6.27 Node judul berhasil dihapus dari file

b. Pengujian tidak normal

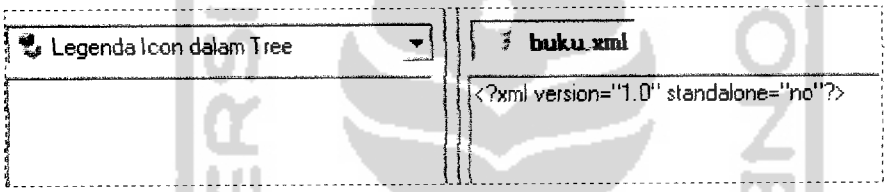
Pengujian ini dilakukan dengan usaha penghapusan node *root*. Sedangkan penghapusan node *root* akan menjadikan file XML tidak lagi wellformed dan valid. *Root*

pada file "buku.xml" adalah node "buku", setelah node ini dihapus maka akan muncul pesan seperti pada gambar 6.25. Setelah penekanan tombol "OK" maka node "buku" akan terhapus sehingga file menjadi tidak wellformed dan memunculkan pesan tidak wellformed seperti pada gambar 6.28.



Gambar 6.28 Pesan tidak wellformed

Setelah itu tampilan akan berubah ke mode teks seperti ditunjukan pada gambar 6.29.

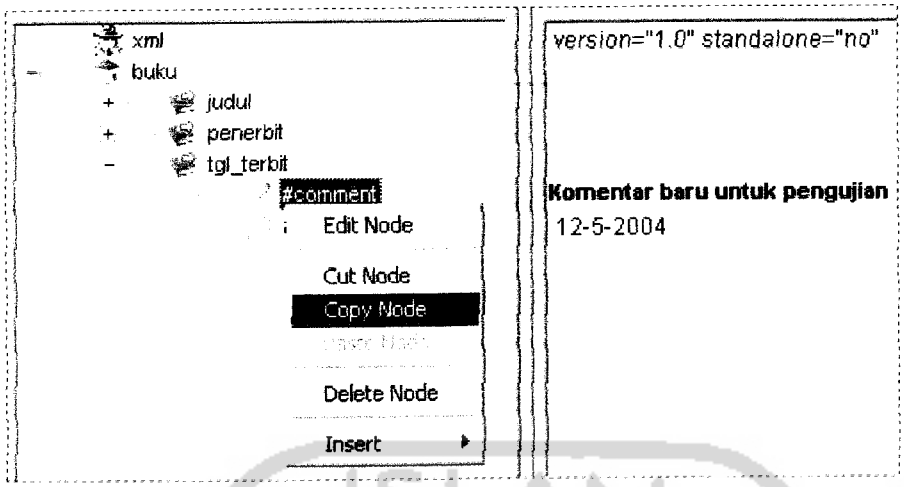


Gambar 6.29 Mode teks untuk file "buku.xml" setelah *root* dihapus

6.5 Pengujian proses cut, copy, dan paste

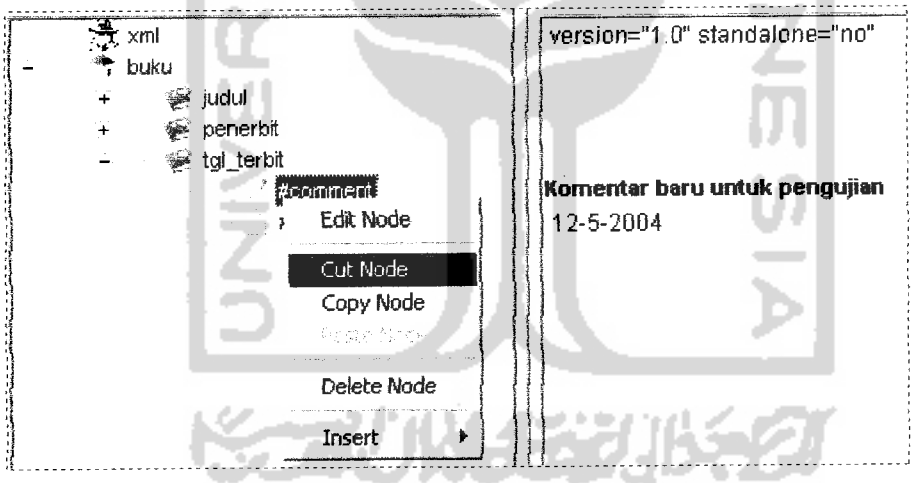
Pengujian ini dilakukan dengan perlakuan yang normal. Pengujian tidak normal tidak dilakukan karena aplikasi tidak memperkenankan proses paste pada tempat yang salah, jadi kemungkinan untuk terjadi kesalahan sangat kecil.

Proses copy dan cut hampir sama, perbedaanya adalah node yang dikenai proses cut akan dihapus sedangkan node yang dikenai proses copy akan tetap ada. Proses paste bertujuan untuk menyisipkan node hasil cut dan copy ke dalam *tree*. Gambar 6.30 akan mengilustrasikan proses copy pada node tgl_terbit| comment.



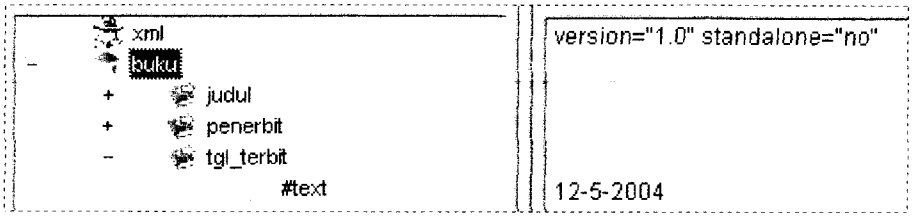
Gambar 6.30 Proses copy node tgl_terbit| comment

Node yang dikenai proses copy tidak mengalami penghapusan. Hal ini berbeda dengan proses cut yang menghapus node yang dikenai proses cut tersebut. Gambar 6.31 akan mengilustrasikan proses cut pada node tgl_terbit| comment.



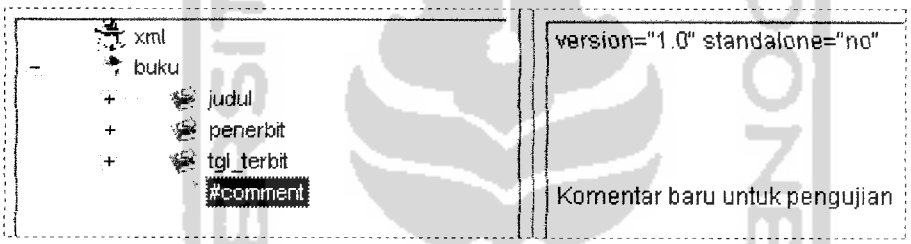
Gambar 6.31 Proses cut node tgl_terbit| comment

Setelah dikenai proses cut, maka node comment akan dihapus dari *tree* seperti terlihat pada gambar 6.32.



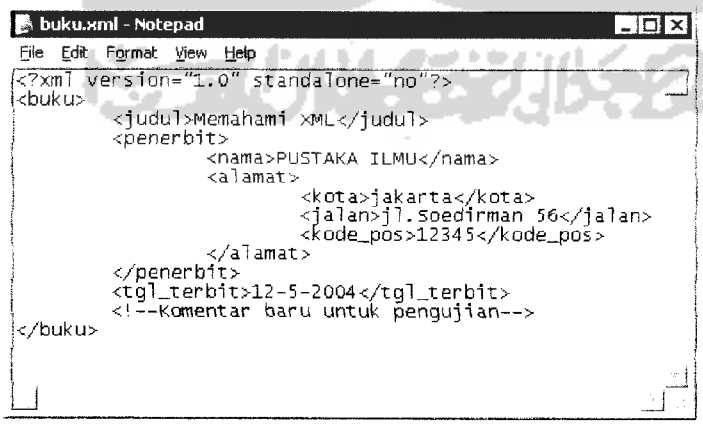
Gambar 6.32 Node tgl_terbit| comment setelah dikenai cut

Proses paste berfungsi untuk menyisipkan node ke dalam *tree*. Node yang disisipkan tersebut berasal dari proses copy maupun paste. Proses paste node comment yang berasal dari proses cut. Node akan disisipkan diantara node judul dan node penerbit. Kemudian node comment akan muncul sebagai child terakhir dari node buku. Gambar 6.33 menunjukan node comment yang sudah di-paste.



Gambar 6.33 Node comment berhasil di paste ke dalam *tree*

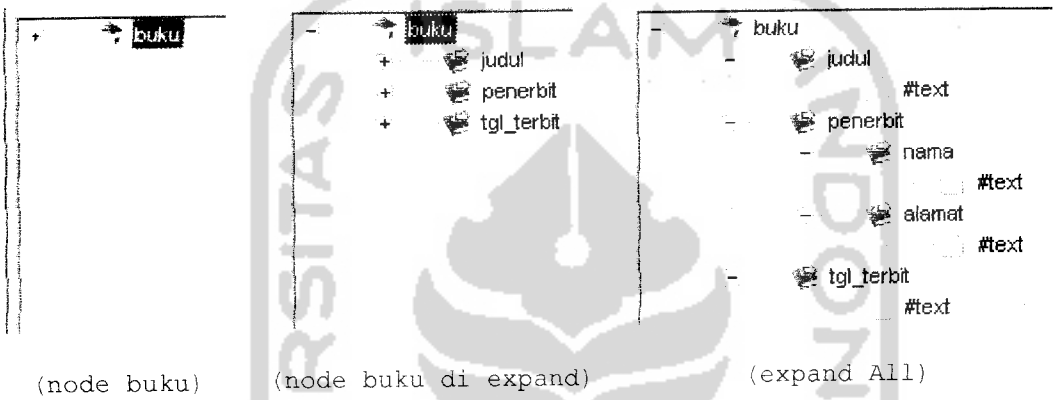
Setelah disimpan maka di dalam file XML akan terjadi penambahan node comment seperti terlihat pada gambar 6.34



Gambar 6.34 Node comment berhasil disisipkan dalam file

6.6 Pengujian proses expand dan collapse

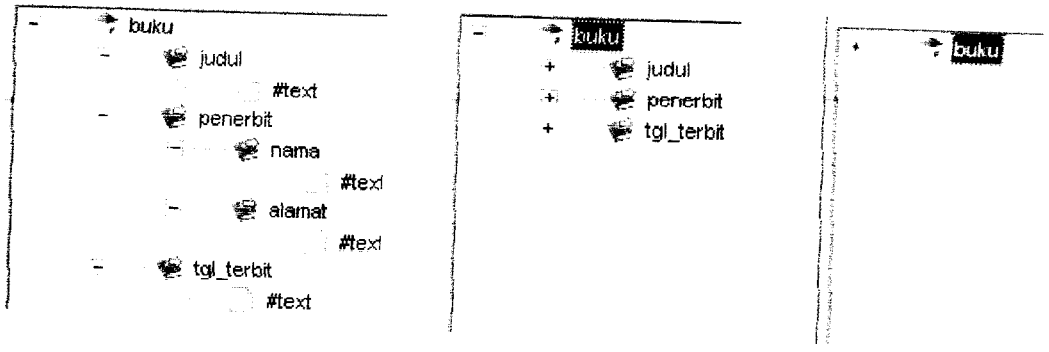
Proses expand bertujuan untuk menampilkan node-node yang terletak di bawah suatu node. Di dalam *tree* node yang mempunyai child ditandai dengan tanda "+", node semacam inilah yang dapat dikenai proses expand. Sedangkan node yang tidak mempunyai child ditandai dengan tanda "-". Pengujian proses expand digambarkan pada gambar 6.35 berikut ini.



Gambar 6.35 Node yang dikenai fungsi expand

Pada gambar pertama terlihat node buku sebelum di-expand. Gambar kedua menunjukan node buku yang sudah di-expand, sehingga terlihat child yang satu tingkat di bawah node buku. Gambar ketiga menunjukan seluruh node dikenai proses expand (*Expand All*).

Proses collapse bertujuan untuk menyembunyikan child dari suatu node. Gambar 6.36 menunjukan pengujian proses collapse.



(node buku) (node judul, penerbit, dan tgl_terbit collapse) (collapse All)

Gambar 6.36 Node yang dikenai fungsi collapse

Terlihat pada gambar pertama di atas node buku dan seluruh node di bawahnya dalam status expand. Gambar kedua menunjukkan node judul, penerbit, dan tgl_terbit dikenai proses collapse sehingga child-nya tidak terlihat lagi.

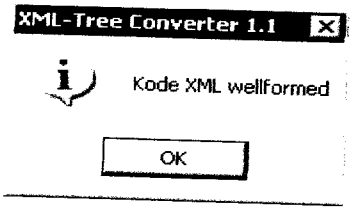
6.7 Pengujian validasi

6.7.1 Pengujian validasi wellformed

Pengujian ini dilakukan untuk mengetahui status wellformed dari file XML (validasi wellformed dijelaskan pada BAB IV sub 2.5.1).

a. Pengujian dengan data normal

Pengujian akan menggunakan file "buku.xml", kemudian akan tampil pesan seperti pada gambar 6.37.

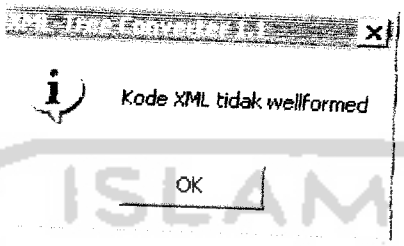


Gambar 6.37 Pesan wellformed

Hal ini berarti file "buku.xml" memenuhi aturan sintaks atau wellformed.

b. Pengujian dengan data tidak normal

Pengujian akan menggunakan file "salah.xml", kemudian akan tampil pesan seperti pada gambar 6.38.



Gambar 6.38 Pesan tidak wellformed

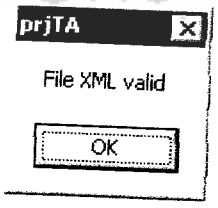
Hal ini berarti file "salah.xml" tidak memenuhi aturan sintaks atau wellformed.

6.7.2 Pengujian validasi valid

Pengujian ini dilakukan untuk mengetahui status valid dari file XML (validasi valid dijelaskan pada BAB IV sub 2.9).

a. Pengujian dengan data normal

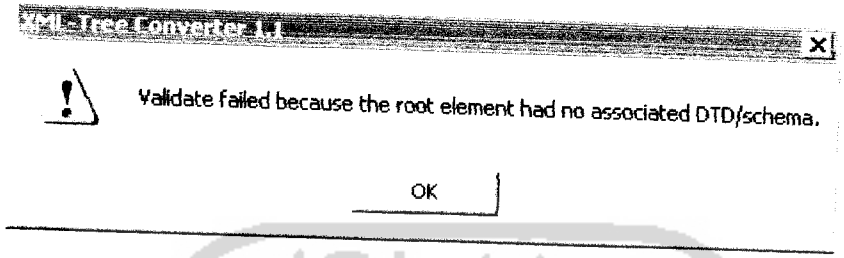
Pengujian akan menggunakan file "node_lengkap.xml", karena di dalam file ini terdapat file DTD internal pada node doctype. Pesan pada gambar 6.39 menunjukan file XML berstatus valid.



Gambar 6.39 Pesan valid

b. Pengujian dengan data tidak normal

Pengujian akan menggunakan file "buku.xml", karena file ini tidak mempunyai file DTD internal ataupun rujukan pada file DTD eksternal pada node doctype. Hal ini akan memicu munculnya pesan seperti pada gambar 6.41.



Gambar 6.40 Pesan tidak valid

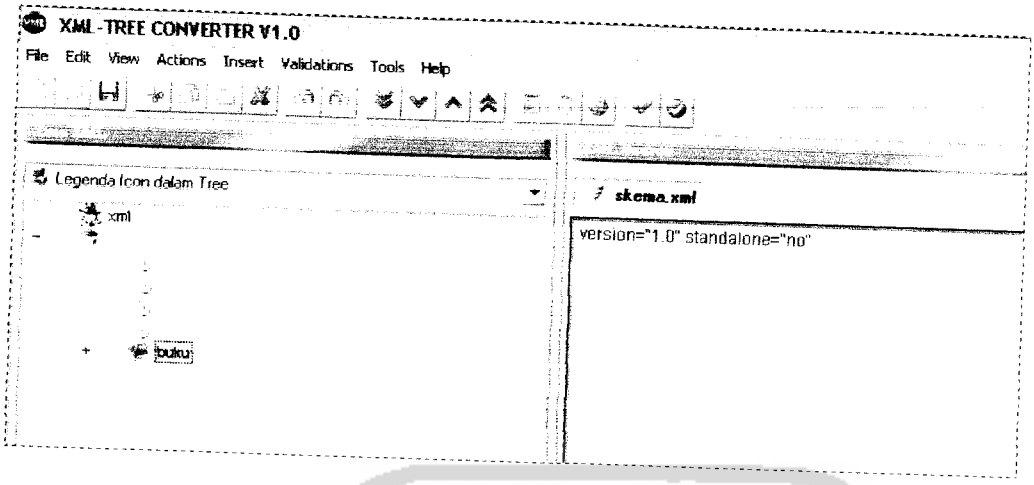
6.7.3 Pengujian skema

Skema merupakan salah satu metode dalam XML untuk mendefinisikan data. Berbeda dengan DTD yang berbentuk tree, skema mengacu pada suatu kerangka besar yang dimiliki suatu vendor di dalam internet yang bisa diacu oleh banyak pengguna produknya. Berikut merupakan pengujian terhadap skema.

Contoh file XML sederhana dengan skema:

```
<?xml version="1.0" standalone="no"?>
<OrgChart xmlns="http://www.xmlspy.com/schemas/orgchart"
xmlns:ipo="http://www.altova.com/IPO"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart
OrgChart.xsd">
     <buku>
     <judul>
     Belajar XML 1.0
     </judul>
     </buku>
</OrgChart>
```

Kemudian setelah dipanggil menggunakan aplikasi akan terlihat sebagai berikut :

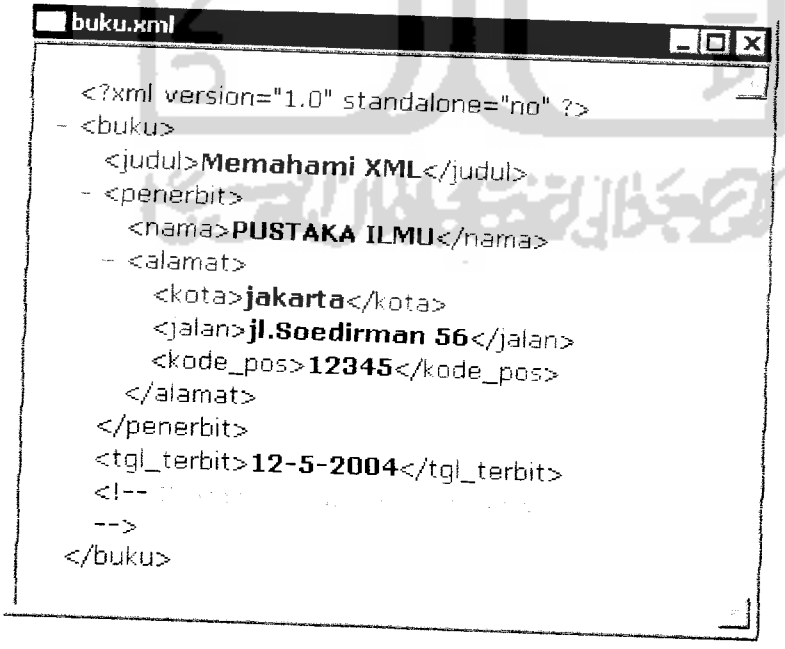


Gambar 6.41 Tampilan skema

Hal ini terjadi karena aplikasi berusaha menghubungkan diri dengan internet namun tidak berhasil.

6.7.4 Pengujian browser

Pengujian ini dilakukan untuk mengetahui tampilan mode *browser* untuk file XML. Pengujian akan menggunakan file "buku.xml" yang telah lulus uji validasi. Gambar 6.42 merupakan tampilan dari file "buku.xml" dalam *browser*.



Gambar 6.42 Tampilan "buku.xml" dalam browser

6.8 Analisis Hasil Pengujian

Pengujian yang dilakukan menghasilkan analisis sebagai berikut:

a. Pengujian proses konversi file XML

Pengujian pertama dilakukan dengan data normal berupa file "node_lengkap.xml". File ini hanya mempunyai 10 jenis node, hal ini dikarenakan node document dan node fragment tidak ikut diuji (lihat sub bab 6.1.1.a). Pengujian ini berhasil mengubah seluruh node di dalam file tersebut dalam tampilan *tree*.

Aplikasi tidak mengubah file XML yang tidak lulus uji validasi ke dalam *tree*, hal ini sesuai dengan pengujian kedua. yang dilakukan dengan data yang tidak wellformed. Sehingga dapat disimpulkan aplikasi hanya mampu mengkonversi file XML yang memenuhi aturan validasi.

Pengujian kedua membuktikan bahwa aplikasi dapat membuat file XML yang baru. File tersebut kemudian dapat ditampilkan di dalam *tree*. File baru tersebut juga dapat disimpan di dalam media penyimpanan dengan penekanan tombol "save".

b. Pengujian proses insert

Pengujian ini melibatkan file "buku.xml". Proses insert yang dilakukan pada node "tgl_terbit| #text" hanya menawarkan insert node element, text, cdata, atau comment (aturan penempatan node dapat dilihat pada BAB II). Hal ini menandakan aplikasi mampu membaca suatu node dan mengidentifikasinya. Sehingga kemungkinan user untuk menambah node pada posisi yang salah dapat diminimalisasi (lihat aturan posisi node BAB 4 sub 4.3).

Parser MSXML 4.0 hanya membolehkan node doctype hanya untuk pembacaan dan validasi sehingga aplikasi tidak dapat menambahkan node doctype dan node yang ada didalamnya yakni node entity dan node notation. Sehingga node yang dapat dikenai proses insert hanya berjumlah 7 jenis node. Pengujian insert nama node yang tidak sesuai aturan penamaan node akan menimbulkan pesan kesalahan.

c. Pengujian proses edit

Pengujian ini berhasil mengubah node "judul" menjadi "judul_buku". Hal ini dikarenakan nama node "judul_buku" sesuai aturan penamaan node (lihat BAB II sub 2.5). Hal ini membuktikan aplikasi mampu mengubah node dalam *tree*. Sedangkan pengujian dengan mengganti node "judul" menjadi node "judul buku" tidak berhasil karena terdapat spasi sehingga tidak sesuai aturan penamaan node.

d. Pengujian proses delete

Pengujian pertama dilakukan dengan menghapus node "judul". Setelah penekanan tombol delete akan memicu pesan yang nenanyakan keyakinan user akan proses delete. Penekanan tombol "OK" menyebabkan data terhapus dari *tree* dan setelah disimpan maka node dalam file akan hilang. Hal ini menandakan aplikasi mampu menghapus node. Pengujian kedua dilakukan dengan menghapus *root* yaitu node "buku". Karena *root* merupakan syarat validasi maka file menjadi tidak wellformed dan valid lagi. Setelah itu aplikasi akan mengubah mode menjadi mode teks. Hal ini menandakan aplikasi mampu menvalidasi ketika terjadi manipulasi dalam *tree*.

e. Pengujian proses copy, cut, dan paste

Pengujian pertama adalah proses copy, setelah node dicopy maka secara otomatis tombol "paste" pada menu konteks akan menjadi *enable*. Node pada *tree* akan tetap ada.

Sedang pada pengujian kedua yaitu proses cut, node akan dihapus dari *tree*. Pengujian terakhir memasukan node ke dalam *tree* melalui proses paste. Hal ini menandakan aplikasi mampu menjalankan proses copy, cut, dan paste dengan baik.

f. Pengujian proses validasi wellformed

Pengujian normal dilakukan dengan mem-validasi file "buku.xml" dan muncul pesan bahwa file tersebut wellformed. Pengujian tidak normal melibatkan file "salah.xml" kemudian muncul bahwa file tersebut tidak wellformed. Sehingga dapat disimpulkan aplikasi mampu membedakan antara file yang wellformed dan yang tidak wellformed.

g. Pengujian proses validasi valid

Pengujian normal dilakukan dengan mem-validasi file "node_lengkap.xml", file ini mempunyai DTD internal. Setelah divalidasi akan muncul pesan valid. Sedangkan untuk pengujian tidak normal menggunakan file "buku.xml" yang tidak memiliki DTD internal dan tidak juga merujuk ke DTD eksternal. Sehingga akan muncul pesan tidak valid. Hal ini membuktikan aplikasi mampu mengidentifikasi file yang valid dan yang tidak valid.

h. Pengujian skema

Pengujian bersifat tidak normal karena skema tidak berhasil di-load ke dalam aplikasi. Hal ini terjadi karena skema mempunyai cara yang berbeda dalam cara kerjanya dengan menggunakan DTD.

i. Pengujian *browser*

File "buku.xml" dapat ditampilkan dalam *browser*, berarti aplikasi mampu menampilkan file XML ke dalam *browser*.

BAB VII

SIMPULAN DAN SARAN

7.1 Simpulan

Hasil dari penelitian ini adalah Sistem Konversi dan Manipulasi File XML versi 1.0. Aplikasi ini dapat bermanfaat untuk membantu pembacaan file XML terutama file XML yang berukuran besar dengan mengubah file tersebut menjadi *tree*. Aplikasi juga dapat melakukan manipulasi dan validasi file XML.

Pengujian telah dilakukan untuk mengetahui kemampuan aplikasi dalam mengubah file XML ke dalam *tree* dan memanipulasinya. Dari hasil pengujian didapat kesimpulan sebagai berikut:

a. XML mempunyai 12 jenis node yang berbeda. Aplikasi hanya mampu menampilkan 10 jenis node ke dalam *tree* dan hanya mampu memanipulasi 7 jenis node. Sedangkan 2 jenis node yang lain hanya digunakan sebagai faktor dalam proses validasi. Hal ini terjadi dikarenakan terbatasnya fasilitas yang diberikan oleh *parser* yang digunakan yaitu MSXML versi 4.0.

b. Fasilitas manipulasi yaitu edit, cut, copy, paste, dan delete untuk node di dalam *tree* dapat digunakan dengan baik. Penggunaan fasilitas ini sangat membantu dikarenakan adanya proses yang selektif. Node *root* yang akan dihapus akan memicu pesan peringatan bahwa penghapusan *root* dapat menyebabkan file XML tidak valid. Ketika akan mem-paste node, aplikasi akan mengenali jenis node yang

akan di-paste. Sehingga proses paste tidak menimbulkan kesalahan sintaks, misal node doctype tidak boleh di-paste di bawah *root*.

c. Proses validasi wellformed yang benar sehingga file XML yang tidak wellformed secara otomatis akan dialihkan ke mode teks.

Secara keseluruhan fasilitas yang tersedia dapat berjalan dengan benar, hanya aplikasi tidak mempunyai kemampuan untuk menampilkan dan memanipulasi seluruh jenis node dalam XML.

7.2 Saran

Beberapa saran untuk pengembangan aplikasi dalam penelitian ini adalah sebagai berikut:

a. Penggunaan versi *parser* MSXMLyang lebih baru sehingga fasilitas yang dibangun dapat lebih bervariasi dan proses manipulasi akan lebih maksimal.

b. Aplikasi diharapkan dilengkapi dengan proses *searching* / pencarian node.

c. Metode validasi valid untuk XML saat ini ada dua jenis yaitu model DTD dan *Schema*, sehingga diharapkan aplikasi mampu mendukung kedua metode validasi tersebut.

d. Pembuatan parser yang dapat digunakan untuk membantu proses manipulasi 12 jenis node di dalam XML

DAFTAR PUSTAKA

[BEN03] Benz, Brian dan John R. Durant. XML Programming Bilble. New York: Wiley Publishing. Inc, 2003.

[COR00] Corporation, Microsoft. Frequently Asked Questions About XML, Microsoft Corporation, 2000.

[COR01] Corporation, Microsoft. MSDN Library. Microsoft Corporation, 2001.

[COR98] Corporation, Microsoft. Enabling Next Generation Web Applications, 1998.

[HAR04] Harold, Elliote Rusty. XML 1.1 Bible 3rd Edition. Indianapolis: Wiley Publishing. Inc, 2004.

[LAN99] Land, Steve. The ASCII of the Future, Microsoft Corporation, 1999.

[PRE02a] Pressman, Roger S. Rekayasa Perangkat Lunak: Pendekatan Praktisi Jilid 1. Terjemahan LN Harnaningrum. Yogyakarta: ANDI, 2002.

[PRE02b] Pressman, Roger S. Rekayasa Perangkat Lunak: Pendekatan Praktisi Jilid 2. Terjemahan LN Harnaningrum. Yogyakarta: ANDI, 2002.

[RAN99] Randell, Brian. A Beginner's Guide to the XML DOM, Microsoft Corporation, 1999.

[RAY03] Ray, Erik T.. Learning XML, California : O'Reilly & Associates, Inc., 2003.

[WYK02] Wyke, R. Allen, Rehman, Sultan, dan Leupen, Brad. XML Programming. Microsoft Corporation, 2002.

[XML01] Altova. XML Spy. Altova Gmbh & Altova Inc.