JUSI Vol. 1, No. 1
Februari 2011

ISSN 2087-8737

Sistem Pemroses Lembar Jawab Komputer Berbasis XML

Arif Rahman

Program Studi Sistem Informasi, Universitas Ahmad Dahlan
Jalan Prof. Dr. Soepomo, S.H, Janturan, Yogyakarta

Email: arif@uad.ac.id

ABSTRAK *Otomatisasi pemrosesan form isian data menggunakan Lembar Jawab Komputer (LJK) dewasa ini diperlukan karena jumlah data yang yang semakin meningkat. Beberapa tahun ini dikembangkan perangkat pemeriksa lembar jawab yang ekonomis namun tidak mengurangi keakuratan pemeriksaan jawaban,yaitu Digital Mark Reader (DMR), yang menggunakan pemindai sebagai pembaca LJK. Pemrosesan data hasil pembacaan pemindai memerlukan prianti lunak khusus. DMR menggunakan format khusus untuk layout form LJK yang hanya bisa dibaca oleh software DMR. Format LJK yang lebih umum dan terstandar dapat disusun menggunakan dokumen eXtensible Markup Language (XML).Dengan format ini diharapkan data hasil pengolahan LJK dapat dimanfaatkan secara lebih luas. Penelitian dilakukan dengan cara mengembangkan suatu sistem pembaca lembar jawab komputer dengan format XML. Sistem terdiri dari hardware dan software. Hardware terdiri dari komputer dan pemindai. Software untuk pembaca lembar jawab secara khusus dibangun. Langkah-langkah yang dilakukan yaitu analisis dan perancangan sistem, pembangunan sistem dan pengujian sistem. Sistem yang dibangun dalam penelitian ini telah dapat membaca lembar jawab komputer hasil pemindaian dengan tingkat akurasi 80%. Tingkat keakuratan ditentukan oleh posisi kertas pada proses pemindaian dan ketebalan arsiran pada bulatan jawaban.*

***Kata kunci**: Lembar Jawab, LJK, DMR, XML*

1 Pendahuluan

Beberapa tahun terakhir ini telah dikembangkan otomatisasi pemrosesan form isian data menggunakan Lembar Jawab Komputer (LJK) atau Dokumen Bertanda Jawab (DBJ). Otomatisasi tersebut diperlukan karena jumlah data atau kuisioner yang besar menyebabkan kebutuhan akan efisiensi pemrosesan mutlak diperlukan. LJK dibaca dan diproses menggunakan piranti yang disebut *Optical Mark Reader* (OMR) yaitu sejenis pemindai khusus yang dapat mendeteksi tanda pada lembar jawab. Tanda tersebut berupa bulatan hitam yang dibuat oleh pengisi jawaban menggunakan pensil, pada umumnya jenis pensil 2B. OMR tersebut dapat mendeteksi letak tanda tersebut untuk kemudian disimpan informasinya dan diproses lebih lanjut, yaitu dapat berupa penghitungan jumlah jawaban tertentu atau dapat juga dilakukan pencocokan jawaban dengan kunci jawaban yang telah tersedia.

Piranti OMR yang beredar dipasaran memiliki kisaran harga yang tinggi sedangkan kebutuhan akan piranti OMR saat ini tidak hanya pada institusi besar, tetapi juga institusi menengah maupun kecil. Biaya pengadaan OMR dapat memberatkan anggaran atau bahkan tidak terjangkau. Berangkat dari keadaan ini muncul ide untuk mengembangkan perangkat pemeriksa lembar jawab yang ekonomis namun tidak mengurangi keakuratan pemeriksaan

jawaban. Perangkat ini dinamakan *Digital Mark Reader* (DMR) yang dikembangkan oleh LPPM-ITB. Perangkat ini menggunakan pemindai sebagai pembaca lembar jawab komputer, yang harganya lebih murah dibandingkan OMR. Untuk pemrosesan data hasil pembacaan pemindai diperlukan prianti lunak khusus.

DMR menggunakan format khusus dalam formulir LJK yang hanya bisa dibaca oleh software DMR. Format LJK yang lebih umum dan terstandar dapat disusun menggunakan dokumen *eXtensible Markup Language* (XML) yang digunakan untuk mendesain lembar kuisioner. XML merupakan suatu bahasa komputer yang mendefinisikan struktur dan isi dokumen dengan cara memecah dokumen menjadi beberapa rangkaian elemen. Tiap elemen mewakili suatu bagian dokumen. Data hasil pengolahan LJK dalam format ini diharapkan dapat dimanfaatkan secara lebih luas.

2 Landasan Teori

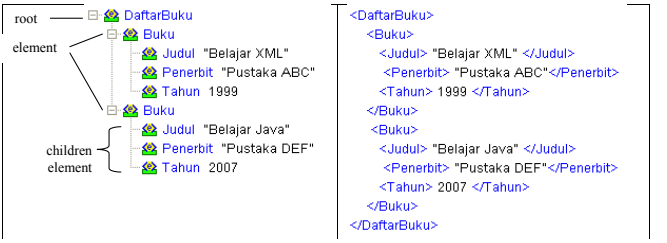
2.1 Digital Mark Reader (DMR)

Digital Mark Reader (DMR) adalah sebuah software yang dikembangkan di Laboratorium Grafika dan Intelegensia Buatan (GAIB) Teknik Informatika ITB sebagai riset unggulan yang didukung oleh LPPM ITB. DMR dikembangkan dengan latar belakang masih sulitnya mendapatkan mesin dan software untuk memproses data ujian pilihan ganda, kuesioner, maupun registrasi yang efisien. Selain itu, harga mesin dan biaya operasionalnya juga masih terhitung cukup mahal. DMR memiliki dua bagian penting, yaitu DMR-Editor (DMR-e) dan DMR Extractor (DMR-x). (Budiarti, 2005)

Pertama-tama Lembar Jawab Komputer (LJK) dirancang pada DMR-e secara visual sesuai kebutuhan. Lalu rancangan tersebut dicetak dan diperbanyak pada kertas. LJK dapat diisi menggunakan pensil, bolpoin, maupun spidol. Setelah diisi, LJK dipindai untuk mendapatkan file citra dari LJK. File citra ini kemudian dibaca oleh DMR sebagai huruf-huruf yang memiliki arti dari LJK yang telah diisi. Proses ini dinamakan ekstraksi. Setelah melalui proses ini, barulah data dapat diolah, antara lain scoring ataupun cetak. Data yang sudah diperoleh dapat diekspor ke format Excel, sehingga data dapat diolah lebih lanjut dengan mudah. (Rahmat dan Supriatna, 2003)

2.2 eXtensible Markup Language (XML)

eXtensible Markup Language (XML) merupakan subset dari *Standard Generalized Markup Language* (SGML) yang didefinisikan oleh World Wide Web Consortium (W3C). XML dikategorikan sebagai *extensible language*, karena memungkinkan pengguna untuk mendefinisikan elemen-elemen *markup* sendiri. Berbeda dengan HTML yang hanya mendefinisikan elemen-elemen untuk tampilan dalam web browser, XML merupakan dokumen multifungsi yang struktur dokumennya mewakili struktur datanya. Secara umum struktur dokumen XML merupakan suatu pohon hierarki, yang terdapat elemen *root*, *parent* dan *child*; seperti ditunjukkan pada Gambar 1.



Gambar 1. Sruktur dokumen XML (a), File XML (b)

2.2.1 Elemen

XML mendefinisikan struktur dan isi dokumen sebagai suatu rangkaian elemen, setiap elemen mewakili suatu bagian dari dokumen (McNeil, 2006). Elemen XML terdiri atas:

(a) Start tag <
(b) Isi atau elemen (dapat berupa teks atau angka)
(c) End tag />

Start tag menunjukkan awal dari suatu elemen dan *end tag* menunjukkan akhir dari suatu elemen. Aturan penulisan nama elemen dalam XML bersifat *case sensitive* dan tidak boleh ada spasi. Tiap elemen XML dapat memiliki satu atau lebih anak yang masing masing anak tersebut juga merupakan suatu elemen.

2.2.2 Atribut

Elemen dalam XML dapat memiliki penjelasan atau keterangan yang disebut dengan atribut. Satu elemen dapat memiliki lebih dari satu atribut. Cara penulisan atribut yaitu dengan menuliskan nama atribut diikuti tanda "=" dan nilai dari atribut tersebut, seperti contoh berikut ini:

```
<DaftarBuku perpustakaan = "Ayo Membaca"  tahun = "2009">
…
…
</ DaftarBuku >
```

2.2.3 Pembacaan Dokumen XML

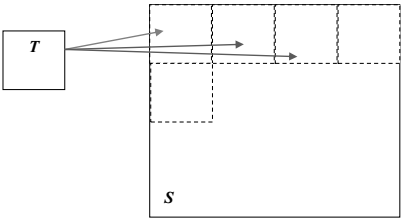
Syarat agar dokumen XML dapat dibaca yaitu harus memenuhi dua level kebenaran, meliputi:

(a) *Well-Formed*: dokumen XML memiliki sintaks yang benar, antara lain kesesuaian penulisan elemen, start-tag (<>) harus diikuti end-tag (</ >)
(b) *Valid*: isi elemen-elemen dalam dokumen XML sesuai dengan aturan semantik, atau tipe data yang dimaksud, misal isi elemen yang tidak terdefinisi nilainya dianggap tidak valid.

Proses pembacaan dokumen XML disebut juga *parsing*. Ada beberapa pustaka atau perangkat lunak untuk membaca file XML, antara lain MSXML dari Microsoft untuk pemrograman di lingkungan Windows dan Xerces untuk pemrograman Java.

2.3 Template Matching

Template matching adalah suatu teknik untuk menemukan suatu bagian dari citra yang cocok dengan citra template atau citra pola yang dicari. Metode sederhana yang umum digunakan adalah *Linear Spatial Filtering*. Langkah pertama yaitu menentukan citra *template* yang akan dicari, dinotasikan sebagai ***T***. Citra tersebut dapat diambil dari citra yang akan dicari atau citra lain. Kemudian citra yang akan dicari kesesuaiannya dengan ***T***, dinotasikan sebagai citra ***S***. Untuk setiap nilai piksel dalam ***S*** atau ***S(x,y)*** akan dicari selisihnya dengan setiap nilai piksel dalam $T(x_t,y_t)$. x,y dan x_t,y_t adalah koordinat posisi piksel (Ballard dan Christopher, 1982). Proses tersebut dapat digambarkan seperti pada Gambar 1.



Gambar 2. Proses *Template Matching*

Selisih antara piksel-piksel dalam ***T*** dan ***S*** dihitung menggunakan SAD (*Sum of Absolute Differences*) atau jumlahan selisih absolut dari masing-masing piksel yang dirumuskan:

$$SAD(x,y) = \sum_{i=0}^{M}\sum_{j=0}^{N}|S(x+i,y+j) - T(i,j)| \quad (1)$$

Citra ***T*** berukuran MxN piksel dan x,y adalah koordinat posisi kiri atas dari bagian citra ***S*** yang dibandingkan. Untuk menentukan bagian ***S*** yang paling cocok dengan ***T*** dicari SAD yang minimal.

$$M = min_{x,y}SAD(x,y) \quad (2)$$

Bagian dengan nilai SAD minimal inilah yang dipilih sebagai bagian yang dianggap cocok dengan *template* yang dicari.

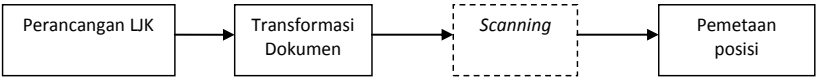
3 Pembahasan

Pengembangan sistem secara garis besar dibagi menjadi tiga tahap, yaitu pra-proses, pembacaan lembar jawab dan pasca-proses.

3.1 Pra-Proses

Pada tahap ini dilakukan perancangan dan pembuatan lembar jawab komputer berbasis XML kemudian mentransformasikannya ke dalam bentuk formulir cetak atau tampilan di layar menggunakan XSLT. Form yang telah dicetak tersebut kemudian dipindai untuk mendapatkan gambar sebagai referensi posisi

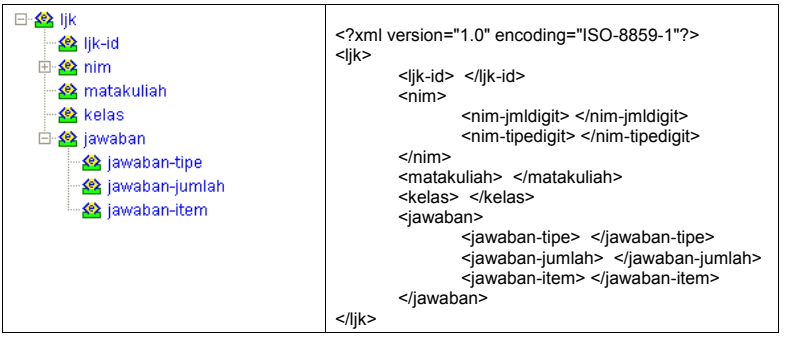
lembar jawab. Langkah selanjutnya yaitu memetakan posisi setiap item jawaban yang nantinya digunakan sebagai dasar pembacaan jawaban. Hasil pemetaan tersebut kemudian disimpan agar dapat digunakan pada tahap berikutnya. Langkah-langkah pada tahap ini dapat digambarkan sebagai berikut.



Gambar 3. Pra-proses

3.1.1 Perancangan LJK

Perancangan lembar jawab menentukan item-item yang akan dimunculkan pada lembar jawab. Format lembar jawab dalam sistem berbentuk file XML dengan elemen-elemennya mewakili seting item-item lembar jawab (Gambar 4).



Gambar 4. Format file XML

File XML dengan format di atas tiap-tiap elemennya memberi informasi kepada sistem sebagai berikut:

Tabel 1. Daftar Elemen XML Lembar Jawab

No	Elemet	Informasi
1	<ljk-id>	Identitas/kode ljk
2	<nim-jmldigit>	Jumlah digit NIM Mahasiswa
3	<nim-tipedigit>	Tipe digit NIM: angka atau huruf
4	<matakuliah>	Kode matakuliah
5	<kelas>	Kode kelas
6	<jawaban-tipe>	Tipe tampilan item jawaban: kotak atau lingkaran
7	<jawaban-jumlah>	Jumlah jawaban
8	<jawaban-item>	Isi jawaban

3.1.2 Transformasi Dokumen

Fokus dari tahapan ini adalah pembuatan tataletak tampilan lembar jawab komputer. Dalam penelitian ini untuk membantu proses tataletak digunakan Microsoft Word®. Ukuran kertas lembar jawab yang digunakan dalam penelitian ini adalah A4. Untuk menampilkan item pilihan berupa bulatan-bulatan, digunakan jenis huruf OMR.

Hasil tataletak tersebut kemudian dikonversi ke dalam format XSL-FO (W3C, 2008) yaitu suatu format *stylesheet* yang dapat digunakan untuk transformasi suatu dokumen ke dalam beberapa macam format dokumen seperti PDF, PS,

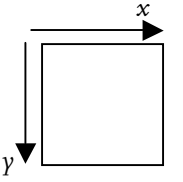
SVG, BMP, PNG, dan lain-lain. Konversi dokumen layout ke format XSL-FO dilakukan menggunakan aplikasi XEP dari RenderX. Tujuan konversi ke bentuk XSL-FO yaitu untuk fleksibilitas format file lembar jawab, sehingga tidak terpaku pada satu bentuk format saja.

3.1.3 Scanning

Pemindaian (*scanning*) pada penelitian ini dilakukan menggunakan perangkat HP Laserjet 1200 Series, dengan resolusi 72 dpi. Proses pemindaian menghasilkan citra bertipe BMP dengan ukuran 503 x 625 piksel. Perangkat pemindai sebaiknya mendukung sistem *document feed* yang dapat melakukan pemindaian beberapa dokumen dalam satu proses.

3.1.4 Pemetaan Posisi

File citra yang didapat pada proses pemindaian akan dipetakan posisi koordinat masing-masing item jawaban. Pemetaan ini diperlukan untuk ketepatan pembacaan dan pencocokan antara hasil pindaian dan jawaban yang telah ditentukan. Ukuran yang digunakan adalah piksel. Posisi dipetakan menggunakan koordinat kartesian dengan ketentuan seperti pada Gambar 5.



Gambar 5. Koordinat kartesian pada citra

Referensi penentuan posisi menggunakan tanda ■ di tepi kiri, kanan atas dan kiri, kanan bawah. Tanda tersebut untuk menjamin pembacaan hasil pemindai tetap lurus walaupun citra hasil pindaian mengalami gangguan posisi. Pembacaan posisi dilakukan dengan menemukan letak tanda ■ kemudian menyimpan posisi *x* dan *y* dari tanda-tanda tersebut sebagai posisi referensi. Kemudian dilakukan pembacaan di seluruh citra untuk mendapatkan posisi masing-masing item jawaban dan menyimpan posisi koordinatnya relatif terhadap posisi referensi yang telah ditentukan. Hasil pemetaan disimpan dalam file teks dengan format sebagai berikut:

```
[nim]
x1:y1 x2:y2 x3:y3 …
[matakuliah]
x1:y1 x2:y2 x3:y3 …
[kelas]
x1:y1
[jawaban]
x1:y1 x1:y2 x1:y3 …
x2:y1 x2:y2 x2:y3 …
…
xn :  yn adalah posisi koordinat pada citra
```

3.2 Pembacaan Lembar Jawab

Pembacaan lembar jawab dilakukan pada file citra hasil pemindaian lembar jawab yang telah diisi. Pembacaan dilakukan dengan cara mencari posisi tanda bulatan-bulatan hitam pada citra. Untuk mengetahui letak bulatan pilihan yang dihitamkan dilakukan langkah-langkah berikut ini:

(a) Konversi format citra hasil pemindaian menjadi format citra skala keabuan (*grayscale*)

Langkah ini dilakukan untuk mendapatkan citra *grayscale* yang terdiri atas 256 level intensitas warna. Ini diperlukan untuk menyederhanakan proses pembacaan selanjutnya.

(b) Konversi citra menjadi citra biner.

Citra *grayscale* yang didapat dikonversi menjadi citra biner untuk menyederhanakan pembacaan letak item jawaban. Konversi dilakukan dengan metode *thresholding* (pemberian nilai ambang batas). Untuk setiap piksel dalam citra, jika nilai piksel ≤ *threshold* maka akan bernilai 0 dan jika nilai piksel ≥ *threshold* maka akan bernilai 1 atau dapat dirumuskan sebagai berikut: jika *p(x,y)* adalah piksel dalam citra ***I***, *q(x,y)* adalah piksel dalam citra biner ***B*** dan ***T*** adalah nilai *threshold*, maka:

$$q(x,y) = \begin{cases} 0, & p(x,y) \le T \\ 1, & p(x,y) > T \end{cases} \quad (3)$$

Nilai *threshold* yang digunakan dalam penelitian ini adalah nilai tengah dari jumlah level intensitas warna yaitu 128.

(c) Pendeteksian letak item jawaban

Deteksi posisi item jawaban berupa bulatan-bulatan yang dihitamkan dilakukan dengan metode *template matching*. Metode ini dapat mencari keberadaan suatu *template* tertentu dalam suatu citra. Pada umumnya ukuran *template* lebih kecil dibandingkan dengan ukuran citra. Pencarian dilakukan dengan mencocokan tiap-tiap wilayah dalam citra dengan ukuran sebesar *template* dan menghitung nilai kecocokannya. Wilayah citra yang memenuhi kriteria nilai kecocokan inilah yang dianggap mewakili posisi keberadaan *template* dalam citra.

Template citra yang digunakan dalam penelitian ini adalah citra bulatan hitam dengan ukuran 10x10 piksel. Citra template dicocokkan dengan tiap-tiap posisi dalam citra yang telah ditentukan berdasarkan peta posisi pada tahap pra-proses. Penghitungan nilai kecocokan (d_T) dilakukan dengan cara menghitung nilai **or** dari tiap piksel dalam *template* (***T***) dan citra (***I***), atau dapat dirumuskan sebagai berikut:

$$d_T = \sum_{x,y=1}^{n} I(x,y) \bigvee T(x,y) \quad (4)$$

$0 \le d_T \le 100$, karena ukuran template adalah 10x10 = 100 piksel

Pada penelitian ini, penentukan nilai kecocokan digunakan *threshold* 0.75 dari total piksel template yaitu 75. Jika $d_T \le 75$ maka dianggap cocok dan jika $d_T \ge 75$ dianggap tidak cocok. Hasil pencocokan *template* item jawaban dengan citra dapat menghasilkan lebih dari satu hasil jika lembar jawab diisi lebih satu bulatan yang dihitamkan, atau tidak ada hasil sama sekali jika tidak ada bulatan

jawaban yang dihitamkan. Jika terjadi kesalahan tersebut maka hasil pembacaan akan diganti dengan karakter '*'. Hasil pembacaan file lembar jawab akan disimpan dalam file teks dengan format sebagai berikut:

```
#XDMR-begin#
nim|mk|kelas|jawab1;jawab2;jawab3…
#XDMR-end#
```

3.3 Pencocokan Jawaban

Item-item jawaban yang berhasil dibaca dari lebar jawab hasil pemindaian akan dicocokkan dengan kunci jawaban yang telah dibuat. Item jawaban berisi data pilihan ganda yaitu salah satu dari A-E. Jika jawaban yang dilingkari lebih dari satu maka hasil baca berupa karakter asterik '*' dan jika jawaban kosong atau tidak dijawab maka hasil baca berupa karakter '-'. Hasil pembacaaan tersebut akan dihitung jumlah benar, salah dan kosong.

4 Percobaan dan Hasil

4.1 Aplikasi Pemroses Lembar Jawab

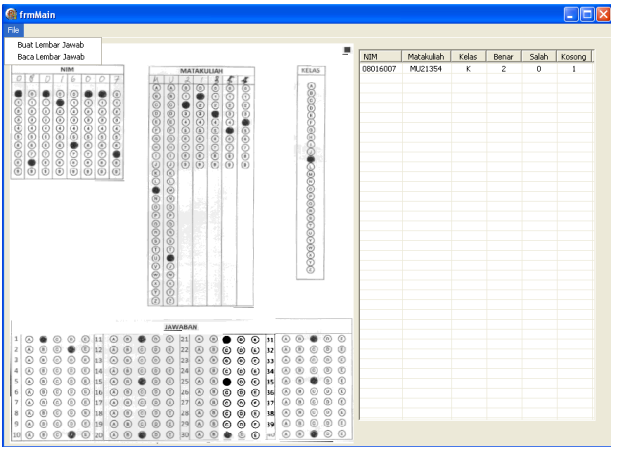
XML-based Digital Mark Reader (XDMR) dikembangkan berdasarkan hasil analisis dan perancangan di atas dengan Delphi 7. Program terdiri atas program utama, pembuat lembar jawab dan pembaca lembar jawab.

(a) Program Utama

Program utama merupakan tampilan utama berisi menu-menu untuk memanggil proses lain yaitu pembuatan lembar jawab dan pembacaan lembar jawab. Tampilan utama menampilkan pula citra hasil pemindaian dan daftar hasil pembacaan lembar jawab dengan format seperti pada Tabel 2. Tampilan program utama seperti pada Gambar 5.

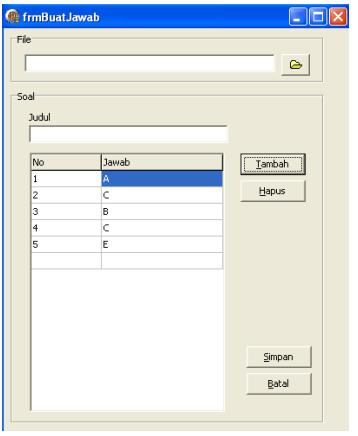
(b) Pembuat Lembar Jawab

Bagian program ini merupakan interface untuk membuat lembar jawab untuk kemudian disimpan ke dalam file XML, dengan nama file sesuai dengan keinginan pengguna. Tampilan pembuat lembar jawab dapat dilihat pada Gambar 6. Pengguna dapat menentukan nama dan letak file pada bagian File. Nama file dapat berupa nama file baru atau nama file yang telah ada. Jika file telah ada maka data dalam file akan ditampilkan.



Gambar 6. Tampilan Program Utama

Pada bagian Soal pengguna dapat memasukkan Judul lembar jawab, dan item-item jawaban. Pengguna juga dapat menambah, mengedit dan menghapus item jawaban, serta menyimpan data keseluruhan lembar jawab. Data Lembar Jawab yang telah diisikan atau diedit akan disimpan ke dalam file XML. Manipulasi file XML dalam Delphi menggunakan komponen TXMLDocument.



Gambar 7. Tampilan Pembuat Lembar Jawab

Kode program untuk membuka file jawaban yang telah ada adalah sebagai berikut:

Listing 1. Membaca File Kunci Jawaban

```
function TfrmBuatJawab.bacaFileJawab(fname :string) :
TStringList;
var
  vNode, vChild : IXMLNode;
  i     : integer;
  lst   : TStringList;
begin
  lst := TStringList.Create;
  XMLDocument1.LoadFromFile(txtNamaFile.Text);
  vNode := XMLDocument1.DocumentElement.ChildNodes['jawaban'];
  vChild := vNode.ChildNodes['item'];
  while  vChild  <> nil do
  begin
     lst.Add(vChild.Text);
```

```
      vChild :=  vChild.NextSibling
  end;
  Result := lst;
end;
```

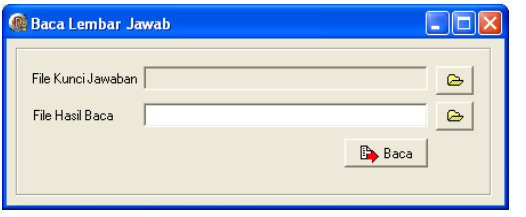
Untuk menyimpan item kunci jawaban, yaitu jika dengan mengklik tombol Simpan maka akan dijalankan program berikut:

Listing 2. Menyimpan Item Jawaban

```
procedure TfrmBuatJawab.btnSimpanClick(Sender: TObject);
var
vNode, vChild : IXMLNode;
i : integer;
begin
   vNode := XMLDocument1.DocumentElement.ChildNodes['jawaban'];
   vNode.Attributes['jumlah']:=
IntToStr(ValueListEditor1.Strings.Count);
   for  i:= 1 to ValueListEditor1.Strings.Count  do
   begin
      vChild := vNode.AddChild('item');
      vChild.Text := ValueListEditor1.Values[IntToStr(i)];
   end;
  XMLDocument1.SaveToFile(txtNamaFile.Text);
end;
```

(c) Pembaca Lembar Jawab

Bagian Pembaca Lembar Jawab merupakan antarmuka pengguna dengan pemindai yang akan membaca lembar jawab dan mencocokkannya dengan kunci lembar jawab yang sesuai. Tampilan Pembaca Lembar Jawab seperti pada Gambar 7.



Gambar 8. Tampilan Pembaca Lembar Jawab

Pengguna memilih file kunci jawaban yang sesuai. File ini merupakan hasil proses pembuatan lembar jawab yang telah dijelaskan pada bagian sebelumnya. Kemudian pengguna memilih file tempat menyimpan file hasil baca Lembar Jawab. File hasil merupakan file teks dengan format yang telah ditentukan, seperti pada Gambar 7. Setelah memilih file-file tersebut untuk melakukan pembacaan lembar jawab melalui pemindai, pengguna mengklik tombol Baca, maka akan muncul pilihan sumber pemindai yang akan digunakan. Selanjutnya proses pemindaian berlangsung dan citra hasilnya akan ditampilkan pada program utama.

Proses pemindaian dilakukan dengan komponen Delphi Twain. Jika tombol Baca diklik akan dijalankan program berikut:

Listing 3. Membaca data dari piranti pemindai

```
procedure TfrmBacaJawab.BitScanClick(Sender: TObject);
var
  SourceIndex: Integer;
  Source: TTwainSource;
begin
DelphiTwain1.LibraryLoaded := TRUE;
DelphiTwain1.SourceManagerLoaded := TRUE;
SourceIndex := DelphiTwain1.SelectSource();
if (SourceIndex <> -1) then
begin
  Source:= DelphiTwain1.Source[SourceIndex];
  Source.Loaded := TRUE;
    Source.SetIPixelType(tbdGray);
    Source.SetIBitDepth(4);
    Source.SetICapUnits(tuPixels);
    Source.SetIYResolution(503);
    Source.SetIYResolution(625);
    Source.SetAutoFeed(TRUE);
    Source.Enabled := TRUE;
 end;
end;
```

Hasil pemindaian berupa citra yang akan dibaca tanda item jawabannya oleh sistem. Bagian-bagian lembar jawab yang dibaca adalah NIM, Matakuliah, Kelas dan Jawaban. Tanda berupa bulatan hitam dalam citra dideteksi menggunakan cara seperti telah dijelaskan di bagian Pembacaan Lembar Jawab. Implementasi pendeteksian mark dalam program adalah sebagai berikut:

Listing 4. Membaca mark

```
function TfrmMain.IsMarked(imBmp,tmpBmp : TBitmap; x,y :
integer) : boolean;
var
i,j, sum : integer;
p : TColor;
pix : byte;
pSource, pRef : integer;

begin
  sum := 0;
  for i:= x to  x + tmpBmp.Width  do
    for j:= y to  y + tmpBmp.Height do
      begin
         p :=  imBmp.Canvas.Pixels[i,j] ;
         pix := Round( 0.299*GetRValue(p)+
                0.587*GetGValue(p) +
                0.114*GetBValue(p) );
         if pix < 128 then pSource := 0
         else pSource := 1;
          p := tmpBmp.Canvas.Pixels[i-x, j-y] ;
         pix := Round( 0.299*GetRValue(p)+
               0.587*GetGValue(p) +
               0.114*GetBValue(p) );
         if pix < 128 then pRef := 0
         else pRef := 1;
         sum := sum + (pSource or pRef);
      end ;
  if sum <= 75 then IsMarked := true
  else   IsMarked := false;
end;
```

Fungsi tersebut mengkonversi tiap piksel dalam citra menjadi piksel bernilai biner, kemudian nilai piksel tersebut di OR-kan dengan nilai piksel dalam citra template. Jika jumlah hasil OR dari tiap piksel ≤ 75 maka dianggap ada tanda atau bernilai *true*, dan jika ≥ 75 maka bernilai *false*.

4.2 Hasil Percobaan

Percobaan dilakukan dengan menjalankan program untuk membaca 10 Lembar Jawab dalam satu jenis lembar jawab dan kunci. Hasil pemrosesan seperti ditunjukkan pada Tabel 2.

Tabel 2. Hasil Percobaan

No	NIM	MK	Kls	B	S	Kosong	Kesesuaian
1	09016001	MU1234	A	5	5	0	Ya
2	09016002	MU1234	A	10	0	0	Ya
3	09016003	MU1234	A	4	6	1	Ya
4	09016004	MU1234	A	7	2	1	Ya
5	09016005	MU1234	A	4	4	2	Ya
6	09016006	0	1	0	0	0	Tidak
7	09016007	MU1234	A	5	4	1	Ya
8	09016008	MU1234	A	3	3	4	Ya
9	09016009	1	0	0	0	0	Tidak
10	09016010	MU1234	A	2	7	1	Ya

Dari tabel hasil percobaan di atas terlihat bahwa terjadi kegagalan pembacaaan formulir pada dua dari 10 data yaitu data ke 6 dan 9. Berdasarkan hasil analisa citra hasil pemindai dapat disimpulkan bahwa kegagalan ini terjadi dikarenakan posisi kertas lembar jawab ketika masuk ke piranti pemindai tidak sempurna, yaitu agak condong ke salah satu arah.

5 Kesimpulan dan Saran

Sistem Pemroses Lembar Jawab Komputer Berbasis XML yang dibangun dalam penelitian ini telah dapat membaca lembar jawab komputer hasil pemindaian dengan tingkat akurasi 80%. Tingkat keakuratan ditentukan oleh posisi kertas pada proses pemindaian dan ketebalan arsiran pada bulatan jawaban.

Tingkat keakuratan sistem ini perlu ditingkatkan terutama mengenai antisipasi kemungkinan posisi pemindaian lembar jawab yang kurang sempurna. Sistem ini sebaiknya juga dilengkapi fasilitas untuk mendesain lembar jawab sendiri.

Referensi

Ballard, D.H. dan Christopher, M.B. (1982), *Computer Vision*, NJ: Prentice-Hall.

Budiarti, A. (2005), *Digital Mark Reader (DMR)*, tersedia di http://ilmukomputer.org/2008/11/25/digital-mark-reader-dmr/ (diakses 21 Desember 2008).

McNeil, V. (2006), *How to Convert a Survey from Word to queXML using Altova XML Spy 2005*, Deakin Computer Assisted Research Facility.

Rahmat, A. dan Supriatna, I. (2003), *Pengantar Digital Mark Reader*, tersedia di http://ilmukomputer.org/2008/11/25/pengantar-digital-mark-reader-dmr/ (diakses 20 Desember 2008).

W3C (2008), *XSL-FO 1.0 Specification*, tersedia di http://www.w3.org/TR/2001/REC-xsl-20011015/ (diakses 20 Juli 2008).

***