KUNGL TEKNISKA HÖGSKOLAN
Royal Institute of Technology
Numerical Analysis and Computing Science

TRITA-NA-D9705 • CID-14, KTH,Stockholm, Sweden 1997

SGML-märkning av medeltida jordeboksmaterial

John Juliusson

CID
Centre for
User Oriented IT Design

John Juliusson
SGML-märkning av medeltida jordeboksmaterial
Report number: TRITA-NA-D9705, CID-14
Publication date: May 97
E-mail of author: john@nada.kth.se
URL of author: http://www.nada.kth.se/~john

Reports can be ordered from:
CID, Centre for User Oriented IT Design
Nada, Dept. Computing Science
KTH, Royal Institute of Technology
S-100 44 Stockhom, Sweden
telephone: + 46 8 790 91 00
fax: + 46 8 790 90 99
e-mail: cid@nada.kth.se
URL: http://www.nada.kth.se/cid/

Innehållsförteckning

Inledning

Elektroniska dokument, kort bakgrund

Efterhand som popularitеten för Word Wide Web växer ökar behovet hos företag och organisationer att distribuera information via Internet. För att göra information tillgängligt på detta medium krävs kunskap om elektronisk publicering, metoder och verktyg. Hur lagras elektroniska texter? Hur presenterar man dem på skärm eller papper? Ett vanligt argument mot digital litteratur är att ingen kommer att vilja läsa en bok från en datorskärm. Det är arbetsamt med kablar, bildskärmar och tangentbord. Vilket förvisso är sant – i dag. Men vi vet lite om hur tekniken ser ut om 10–20 år, det pågår forskning om portabla papperstunna skärmar, som kan komma att förändra denna begränsning. I dag ligger dock datorernas styrka för t.ex. den historiska forskningen i bl.a. förmågan att söka och hantera stora textmassor. En dator kan söka information och jämföra olika material betydligt snabbare än vad någon forskare klarar av manuellt.

Hur går man då till väga för att lagra dokument av allehanda slag elektroniskt? Ett par vanliga krav man har är att dokumenten inte ska vara beroende av någon speciell ordbehandlare eller system för att kunna läsas, och ofta vill man kunna söka och ev. också plocka ut viss information ur ett dokument. **SGML** (Standard Generalized Markup Language, ISO 8879:1986) har visat sig ha många fördelar som lagringsformat. Det är en väl etablerad internationell standard och man bevarar dokuments struktur, det är också ofta konverterbart till en mängd andra format.

Beskrivande märkning och SGML

Beskrivande märkning innebär att man kodar ett dokuments innehåll, t.ex. rubrik, brödtext, författare med mera. Till skillnad från det traditionella sättet som i dag används i de flesta ordbehandlingsprogam, s.k. **procedurell märkning**, som innebär att man lagrar information om dokumentets utseende, fet stil, upphöjd stil m.m., tillsammans med dokumenttexten. Genom att använda beskrivande märkning kan dokumentet användas som en databas. Det går att söka i texten, ställa frågor och plocka ut intressant information.

SGML bygger på beskrivande märkning. Här kodas dokumentet med märkord som kännetecknas av att de omges av tecknen < och />. SGML är en internationell standard för att beskriva dokument eller informations-strukturer, och alla i dag existerande former av information kan kan mer eller mindre detaljerat beskrivas med SGML.

Digital litteratur, TEI

Digital litteratur är än så länge ganska ovanligt inom Sverige, en viss ökning har dock skett med Internet's framväxt. Det lagringsformat som då oftast använts är HTML, Hypertext Markup Languge. Dessa utgåvor har dock en viss brist då de märkord som används i HTML endast beskriver den grafiska presentationen av texten.

TEI är ett stort internationellt projekt genom vilket forskare försöker åstadkomma gemensamma riktlinjer för avancerad kodning av all slags litterära texter och historiska källor. Det har engagerat litteraturvetare, lingvister, historiker och dataloger över hela världen. Projektet har resulterat i en TEI-DTD[1] och ett 700 sidor tjockt kompendium. Detta kompendium kan ses som en anvisning för hur TEI ska tillämpas på olika slags texter. Även en enklare variant av TEIs DTD, TeiLite har tagits fram inom detta projekt.

På många håll i världen finns det databaser innehållandes TEI-kodad humanistisk litteratur av olika slag. En del av dessa är tillgängliga via Internet, t.ex. American Verse Project [21]. I dessa databaser kan man med hjälp av någon SGML-läsare (t.ex. Panorama) läsa de TEI-kodade texterna "online". Det går även att hämta hem och spara texter i sin egen dator från databasen, studera texten noggrannare, och i princip använda materialet hur man vill. I Sverige finns det s.k. Runebergs-projektet, ett projekt som bygger på frivilliga insatser. Här hittar man allehanda svensk litteratur i HTML-format. Men HTML har en del begränsningar, då man endast har tillgång till en begränsad uppsättning märkord. Det vore önskvärt att i Sverige pröva möjligheterna att sätta upp en databas med SGML-kodad text enligt TEIs principer.

Problem

Riksantikvarieämbetet ger ut en skriftserie "Det medeltida Sverige" som innehåller det s.k. jordeboksmaterialet d.v.s. information från medeltiden och 1500-talet om svensk mark, jordens brukning, ägandeförhållanden med mera. De består av excerpter från samtida källor. För vissa landskap finns i dag tryckta utgåvor. Dessa böcker är väldigt kompakt skrivna (många förkortningar) och har en mycket speciell struktur. De är svårlästa för en vanlig amatör. Riksantikvarieämbetet har uttryckt en önskan om att lagra dessa böcker på dator för att göra dem tillgängliga och sökbara via t.ex. WWW (World Wide Web).

Med en väl utförd SGML-märkning kan många olika grupper ha glädje av detta material i elektronisk form. Uppgiften för detta examensarbete har bestått av tre olika delar:

- undersöka befintliga samt kommande standarder inom området elektronisk publicering,
- tillämpa SGML med lämplig DTD (TEI, TeiLite eller annan) på jordeboksmaterialet,
- ta fram en prototyp för hur en användare kan hantera en samling av TEI-märkta dokument. Dessa TEI-dokument kan t.ex. vara dikter, poesi o.s.v. – de behöver alltså inte vara just jordeboksmaterial.

[1]Dokument Type Definition, se nedan under rubriken Principer för SGML

För den andra delen, d.v.s. att tillämpa SGML på jordeboks-materialet, har det uppstått diverse olika delproblem. Bland annat att finna lämpliga datorverktyg för SGML-märkning och konvertering.

Detta exjobb behandlar inte problemen rörande hur dokumenten organiseras och lagras då antalet dokument blir mycket stort, det vill säga hur en TEI-databas ska designas.

Märkningstandarder SGML m.fl.

Principerna för SGML

SGML (ISO 8879:1986) är en internationell standard för att beskriva dokument eller informationsstrukturer. Alla i dag existerande former av information kan mer eller mindre ingående beskrivas med SGML. Dokument som lagras i SGML är läsbara på i dag vanliga datorer och operativsystem (Windows/Dos, Unix, Macintosh). Detta plattformsoberoende är en av grundtankarna med SGML. En annan grundtanke är att dokument som lagras i SGML ska hålla sig ”färska”, det vill säga att de ska kunna läsas även om tio år. Jämför med ett dokument som är lagrat i något populärt ordbehandlingsprogram för tio år sedan, förmodligen måste detta dokument genomgå diverse olika konverteringsfilter innan det kan läsas.

Med SGML skapar man mallar för olika typer av dokument. Dessa dokumentmallar definieras med hjälp av en DTD. Det finns i dag färdiga DTD:er för de flesta områden: förlagsvärlden, elektronik, medicin, rapporter, nyheter med mera. Somliga av dessa är registrerade ISO-standarder, medan andra endast är allmänt vedertagna inom vissa områden. Den DTD som ska studeras närmare i detta projekt (TEIs DTD) är ett exempel på det senare fallet.

SGML innehåller mycket mer än vad som beskrivs i detta avsnitt. För er som vill ha mer information om SGML hänvisas till litteraturlistan [4], [5]. Senare i denna rapport ges även exempel på SGML-märkningar.

Ett SGML-dokuments tre beståndsdelar; SGML-deklaration, DTD och dokumentinstans

Varje SGML-dokument består av tre delar. Dessa delar kan var och en vara uppdelade på olika filer (vanligaste fallet), men de kan också alla finnas i samma fil. Alla dessa tre delar är skrivna i ASCII-text, och de måste förekomma i följande uppräknad ordning:

- **SGML-deklarationen.** I denna finns stora möjligheter att göra lokala anpassningar. Man kan välja teckenuppsättning, vilka nyckelord som ska användas, vilken syntax som ska gälla med mera.
- **DTD, Document Type Definition**. I DTD:n bestäms på ett entydigt sätt dokumentets struktur. Här deklareras alla element, deras tillåtna innehåll, och deras attribut. Här definieras även entiteter och s.k. **short references** (används sällan).

- **Dokumentinstans**. Denna tredje och sista del innehåller själva texten. Dokumentet är uppmärkt med märkord enligt de regler som är angivna i DTD:n.

Element, attribut och entiteter

Ett SGML-dokument vanligaste innehåll är **element, attribut** och **entiteter**. Det finns ytterligare några markeringar som kan förekomma i ett dokument, men de tas inte upp här.

- Element. Ett SGML-dokuments grundläggande byggstenar är elementen. Ett element omges av märkord och kan bestå antingen av ren text, andra element eller en kombination av dessa. All text i ett SGML-dokument måste var inkapslad med märkord, d.v.s. allt i ett SGML-dokument är ett element av någon typ.
- Attribut. Till varje element finns också möjlighet att knyta ett eller flera attribut. Attributet anger egenskaper hos elementet.
- Entiteter. Det finns olika typer av entiteter i SGML. De används i huvudsak till substitution av särskilda tecken, text och filer.

HTML vs. SGML

HTML är en tillämpning av SGML. En något haltande jämförelse kan göras med att ett C-program som räknar ut 1+2 är en tillämpning av programspråket C. HTML motsvaras av programmet som räknar ut 1+2 medan SGML motsvarar programspråket C. HTML beskrivs med andra ord av en DTD som är konstruerad i språket SGML.

Denna HTML-DTD standardiseras genom HTML Working Group och den förändras kontinuerligt. I dag är det HTML 3.2 som är den senaste versionen av HTML-DTD:n. Men det florerar också en del HTML-versioner som inte tillkommit på ”godkänd” väg. Bland annat så har många programvarutillverkare lanserat egna HTML-versioner. Ett exempel på märkord som inte tillhör standarden är Java-applets (`<applet>`).

Relaterade standarder: XML, DSSSL

I detta kapitel ges en kort sammanfattning av en kommande standard, XML, och en relativt ny standard, DSSSL. Båda dessa har nära relation till SGML.

XML – Extensive Markup Language

XML är en standard under arbete. Det kommer periodvis olika utkast från W3C (World Wide Web Consortium), som är ansvarig för standarden. Det är en mycket enkel dialekt av SGML. XML har plockat bort en hel del av de tillval och avancerade egenskaper som sällan används i SGML. Med denna åtgärd hoppas man att XML ska få större spridning över Internet än vad SGML har fått. HTML är begränsat i den mening att man bara kan beskriva en typ av dokument med HTML. I XML (och SGML) kan man man beskriva oändligt många dokumenttyper, vilket medför avsevärda förbättringar för Internet. Med XML kan WWW

användas för utbyte av mer komplex information än vad HTML erbjuder.

I dag finns ett dussin olika versioner av HTML-DTDn. HTML-läsare klarar i allmänhet inte av alla dessa olika DTD:er, vilket medför att datorn låser sig då man surfar och kommer till en sida med inkompatibel HTML-DTD. Behovet av XML har dykt upp nu när HTML har några år på nacken, och man har upptäckt att det blir väldigt rörigt med en mängd olika HTML-DTD:er. Hur XML kommer att utvecklas och vad det får för konsekvenser för Internet är en fråga som är intressant.

TEI använder i dag en del SGML-tillägg som inte finns i XML. Den dag XML blir en färdig standard kommer troligtvis en XML-version av TEIs DTD att utvecklas.

DSSSL

SGML är i dagsläget varken tänkt eller lämpat som presentationsformat, utan beskriver endast strukturen hos ett dokument. DSSSL (Document Style Semantics and Specifikation Lanugage, ISO 10179:1996) är tänkt att bli en komplettering till SGML, i den mening att den beskriver hur ett SGML-dokument typografiskt ska presenteras på skärm eller papper. Den presentation som finns i dagens SGML-läsare är ”hemsnickrade” varianter och använder inte DSSSL-stylesheet.

DSSSL är en ett år gammal standard. Det består av två olika beskrivande språk: dels ett stilspråk för att beskriva hur ett SGML-dokument typografiskt ska presenteras, och dels ett konverteringsspråk STTP (SGML Tree Transformation Process). STTP är tänkt att användas för att konvertera dels från en DTD till en annan DTD och dels från en DTD till något annat utmatningsformat. Då denna standard är så pass ny så finns mig veterligen endast en tillämpning av DSSSL. Det är konverteringsprogrammet Jade (se om SGML-verktyg nedan), som konverterar med hjälp av DSSSL stylesheets. Ett stylesheet beskriver hur de olika elementen i ett SGML-dokument ska presenteras på skärm eller papper. I dag finns inga program som ”interaktivt” presenterar ett dokument enligt ett DSSSL stylesheet. Men troligtvis kommer detta att dyka upp på marknaden så småningom.

SGML-verktyg

Editor, DTD-konstruktion

Det viktigaste redskapet då man märker ut ett SGML-dokument är själva editorn. Ett par viktiga krav på en editor är i detta fall:

- ska kunna hantera stora filer
- stödja TEIs DTD
- ge stöd vid insättning av märkord

Emacs visade sig vara en lämplig editor som uppfyller ovanstående krav. Det finns en SGML-mode i Emacs som ger stöd åt uppmärkning av dokument. Man kan interaktivt ändra i en DTD, analysera om den och sedan märka upp och validera sitt dokument enligt den nya DTD:n. En annan fördel med Emacs är möjligheten till avancerade sök/ersätt funktioner, vilket ofta kommer till användning vid hantering av texter.

Många av dagens SGML-editorer innehåller begränsningar vad gäller storlek på filer, och ofta är de inte 100 % kompatibla med SGML-standarden. Author/Editor och Near and Far är två kommersiella SGML-editorer som undersökts i detta exjobb. Ingen av dessa klarar dock av TEIs DTD vilket Emacs gör. Då jag har erfarenhet av Emacs sedan tidigare så det föll sig naturligt att använda den. Emacs är dock inte att rekommendera för en nybörjare, då den inte har något standardiserat vänligt gränssnitt. För att använda en mer ”användarvänlig” editor ihop med TEI så krävs att man gör en specialversion av sin färdiga TEI-DTD. För Author/Editor måste man även kompilera om sin DTD till ett speciellt format.

En SGML-validator kontrollerar att ett SGML-märkt dokument följer de regler som är angivna i DTD:n. En validator som används på många håll i världen är Nsgmls, ett fritt program som kan anropas interaktivt ”inifrån” Emacs. Denna validator har även använts i detta projekt.

Konverteringsverktyg

En av fördelarna med SGML är att det anses vara neutralt. Det vill säga det är smidigt att konvertera till många andra vanliga format HTML, PDF (Portable Document Format), RTF (Rich Text Format), och TEX. Det finns en del fria konverteringsprogram att välja mellan, bland annat ett Perl-paket kallat ”Sgmlspm” som används ihop med validatorn och SGML-parsern Nsgmls. Det är ett klassbibliotek som tillhandahåller funktioner för konvertering av SGML-dokument. Med detta Perl-paket kan man relativt enkelt konvertera SGML-dokument dels från en DTD till en annan DTD, och dels från en DTD till något annat format. Programmet kräver dock en viss insikt i Perlprogrammering för att göra mer avancerade konverteringar. Mina erfarenheter av detta konverteringsprogram är goda.

Det finns ett program, Jade, som konverterar med hjälp av DSSSL-stylesheets (se ovan rubriken DSSSL). Ett DSSSL-stylesheet är en beskrivning av hur de olika elementen i en DTD ska presenteras på skärmen. På Internet finns arkiv med stylesheets för ofta använda DTD:er såsom docBook, TEI och HTML. I dagsläget går det att konvertera till RTF, HTML och TEX med Jade. Men i framtiden kommer det säkerligen att vara möjligt att konvertera till flera andra format. Jade finns i Windows95/NT och Unix-miljö.

En inte ovanlig situation är att de dokument som ska SGML-märkas levereras på ett annat format, t.ex. MS Word, RTF eller FrameMaker. I dessa fall finns det diverse olika programvaror för konvertering till önskad SGML-DTD. Programmet Rtf2html konverterar RTF till HTML-DTD. Rainbow är ett annat program som konverterar RTF till en speciell Rainbow-DTD. Med hjälp av ovan nämnda Perl-paket kan sedan konvertering ske till den SGML-DTD man vill ha.

SGML-läsare

I dagsläget är det ont om SGML-läsare som fungerar via WWW. Panorama, MultiDocPro och SplitVision är tre SGML-läsare som studerats närmare. Funktionaliteten i dessa tre program är i stort sett samma, man kan läsa sitt SGML-dokument, byta stylesheet och navigera i dokumentet med en navigator. Den enda av dessa som fungerar via WWW är Panorama. De andra två fungerar än

så länge bara för SGML-dokument lagrade på den lokala maskinen.

Teknisk beskrivning av TEIs DTD

TEI anger riktlinjer för hur humanistisk/historisk litteratur ska kodas enligt SGML. TEIs DTD innehåller i sitt grundutförande ca. 160 märkord. Den är ganska komplex och utnyttjar SGML fullt ut. Detta kan ibland vara ett problem ty många av de SGML-programvaror som i dag finns klarar ännu inte av TEIs DTD p.g.a. detta. Men det är också en styrka för DTD:n utnyttjar vissa av fördelarna med SGML, den blir mer modulärt uppbyggd och det är enklare att modifiera den. TEI tillhandahåller även en bantad version av sin DTD kallad TeiLite.

TEIs DTD är uppdelad på ca. 60 olika filer innehållande entitets- och elementdeklarationer. DTD:n är komplex att sätta sig in i. Men det krävs att ha bra kontroll över dess uppbyggnad då man planerar att modifiera den.

Beskrivning av TEIs DTD finns i ett ca. 700 sidor tjockt kompendium. I detta kapitel ges en något kompaktare sammanfattning, element som är viktiga för strukturen, element använda i JB och hur DTD:n ska läsas kommer här att beskrivas. För att fullständigt förstå detta avsnitt krävs mer kunskap om SGML än vad som tidigare nämnts. Mer ingående information om SGML återfinns i böckerna i litteraturlistan [4], [5].

Strukturen hos TEIs DTD

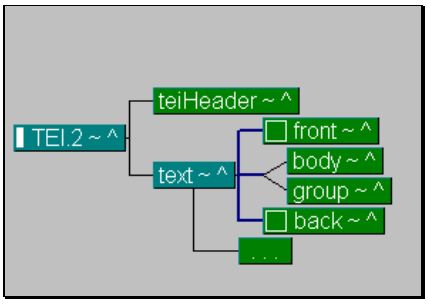
DTD:n är uppdelad i olika byggblock som innehåller olika märkordsuppsättningar. Dessa fyra byggblock är:

- core tag-sets, standardkomponenter (element) som alltid inkluderas
- base tag-sets, grundbyggblock för olika texttyper (ex. vers, prosa, drama), exakt en av dessa måste väljas
- additional tag-sets, utökade märkordsuppsättningar för skilda ändamål (t.ex. figures, names)
- egna skräddarsydda användardefinierade märkord

En minimal TEI-DTD (core + prose tag-set) innehåller ca. 160 märkord. Författarna bakom TEI gör en liknelse med en pizza-modell där man har en deg (core tag-set) i grunden och sedan fyller på med olika märkords-uppsättningar (tomatsås-base, fyllning-additional, och kryddor-egna märkord) beroende på vilken typ av text som ska märkas ut. Denna teknik med att endast ta med de märkords-uppsättningar man är intresserad av åstadkommes med SGMLs **marked sections**.

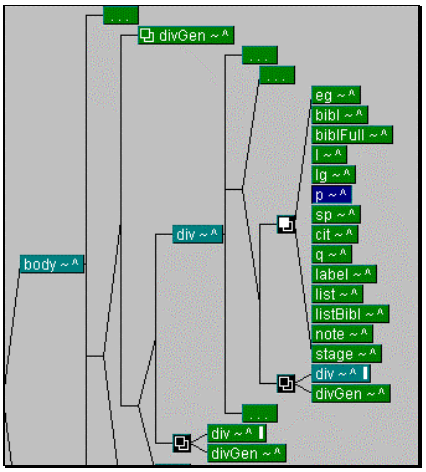
SGML-träd

Nedan finns ett SGML-träd som beskriver grundstrukturen av DTD:n. Lägg märke till att trädet inte är fullständigt, utan bara innehåller själva grundstrukturen av TEI.



Figur 1 TEI DTD - SGML-träd

I teiheadern finns meta-information om dokumentet, utgivare, ISBN-nr, version, filens storlek med mera. Elementet `<text>` innehåller själva texten som är uppdelad i fyra element, varav det viktiga är `<body>`, som i sin tur är uppdelad i div-element, vilka används vid indelning av dokumentet i olika avsnitt.



Figur 2. Elementet `<div>`

På bilden till vänster syns innehållet i elementet `<div>`.. I `<div>` finns de s.k. grupp-elementen (**chunks**, se nedan) eller andra div-element. Inom dessa gruppelement får förekomma s.k. fraselement (**phrase**, se nedan) eller blandelement (**inter-level**, se nedan). Inom vissa av dessa fraselement får sedan förekomma grupp-element. Detta gör att det ibland kan bli något förvirrande att läsa TEIs DTD.

M-klasser och A-klasser

För att få en modulär och strukturerad uppbyggnad av TEIs DTD delas element och attribut in i klasser. Dessa klasser är M-klasser (modelklass) och A-klasser (attributklass). En M-klass grupperar element som är semantiskt lika, d.v.s. som har liknade innebörd och kan förekomma på liknande ställe i ett dokuments struktur. M-klasser används sedan i innehållsmallen för andra element. En M-klass definieras med hjälp av parameterentiteter:

```
<ENTITY % x.date ‘’>
<ENTITY % m.date ‘%x.date date | dateRange | dateStruct’  >
```

M-klassen `m.date` grupperar de tre elementen date, dateRange, och dateStruct. I alla element där `m.date` finns med i innehålls-mallen kan något av dessa tre element förekomma. Entiteten `x.date` används då man behöver lägga till element till en M-klass. Detta förklaras mer utförligt nedan under rubriken ”Att modifiera TEIs DTD”.

A-klasser används på ett liknande sätt för att gruppera attri-but. Till exempel A-klassen `a.names` grupperar de attribut som hör till elementen `<name>`, `<placeName>` och `<persName>`.

```
<!ENTITY % a.names '
reg          CDATA          #IMPLIED
key          CDATA          #IMPLIED'>

<!ELEMENT name       - -     (%phrase.seq;) >

<!ATTLIST name               %a.global;
                             %a.names;
                             type CDATA            #IMPLIED
      >
```

Ovan ses ett exempel på hur A-klasser används. Entiteten `a.names` deklareras med värdet mellan apostroferna. I attributlistan för elementet `name` ersätts sedan entiten `a.names` med dess värde.

Indelning av M-klasser, standard innehållsmallar

M-klasser delas in i två huvugrupper, högnivåklasser och lågnivåklasser. Högnivåklassen delar in element efter följande tre olika elementtyper:

1. gruppelement (chunks), förekommer inom div-element, ex. `<p>`, `<l>`
2. fraselement (phrase), förekommer inom gruppelement, ex. `<date>` (datum), `<abbr>` (förkortning)
3. blandelement (inter-level), får förekomma inom både 1) div- och 2) gruppelement

Lågnivåklasser grupperar element som är strukturellt eller semantiskt lika. Under ovanstående punkt 2 finns det ca. 10 olika lågnivåklasser t.ex. `m.date`, `m.edit`, `m.loc` med mera. På punkt 3 finns fyra lågnivåklasser för blandelement. För punkt 1 finns få element så här behövs inga M-klasser, här finns tolv olika gruppelement.

Ett elements innehåll definieras normalt med någon av ovanstående 3 punkter, d.v.s. det kan innehålla grupp-, fras- eller blandelement. Det finns 6 standard innehållsmallar: `phrase`, `phrase.seq`, `component`, `component.seq`, `paraContent`, `specialPara`, som bygger på denna indelning. Dessa rekommenderas för användning i innehållsmallen då man definierar sina egna element. Detta för att få en konsekvens vid definiering av element.

Att modifiera TEIs DTD

Att modifiera TEIs DTD görs i två steg: 1) definiera nytt element, och 2) bestämma var elementet får placeras inom TEIs dokumentstruktur. Dessa ändringar av TEIs DTD görs i två olika filer, som associeras med parameterentiterna: `TEI.extensions.ent` och `TEI.extensions.dtd`. Dessa två entiteter definieras i doctype-subset deklarationen, t.ex. så här:

```
<ENTITY % TEI.extensions.dtd       SYSTEM  'mitt-projekt.dtd'>
<ENTITY % TEI.extensions.ent       SYSTEM  'mitt-projekt.ent'>
```

I `TEI.extensions.dtd` definerar man egna element, samt gör modifieringar av befintliga element och ändrar och lägger till attribut för element. Detta görs enligt normal SGML elementdeklaration.

I `TEI.extensions.ent` definieras ändringar av TEIs parameterentiteter. Detta innebär att man kan definiera egna M-klasser samt lägga till nya element till redan existerande M-klasser. Att lägga till ett nytt element till en M-klass görs genom att omdefiniera entiteten `x.klass` för godtycklig klass. Genom att

lägga till ett element till en M-klass bestäms i praktiken var någonstans elementet får förekomma, t.ex.:

```
<ENTITY % x.date ‘year | ‘          >
<ELEMENT year - - (%phrase.seq;)    >
```

Det nya elementet `year` får nu förekomma i dokumentet hos alla element där `m.date` finns i innehållsmallen. Elementets (`<year>`) innehållsmall är `(%phrase.seq;)`, vilket betyder att elementet får innehålla en följd av fraselement.

Dokumentanalys, märkningsarbete

Dokumentanalys

Ett av de tidigare problemen i föreliggande arbete var att undersöka hur väl TEIs DTD lämpade sig för JB (Jordeboksmaterialet). Krav på märkningen var följande:

- bevara de publicerade volymernas struktur
- det ska gå att plocka ut detaljerad information genom sökningar

De tre alternativ som finns var att använda TEIs fullständiga DTD, TeiLite DTD:n (en betydligt enklare variant av TEIs DTD), eller att konstruera en egen DTD. TeiLite är inte modifierbar vilket gjorde att vi kunde utesluta den ganska snabbt. Att skriva en egen skräddarsydd DTD för just JB kunde vara en lösning, men det tar ofta mycket tid att utveckla en ny DTD. I all litteratur om SGML ges alltid rådet att om möjligt använda en redan befintlig DTD. Så valet föll på TEIs DTD. Den kanske inte är optimal för JB, man får med en del märkord på köpet som man skulle klarat sig utan. Men tack vare att den är mycket modifierbar och ej heller särskilt tvingande i sin uppbyggnad visade den sig vara lämplig för detta ändamål.

Vilka märkordsuppsättningar behövs?

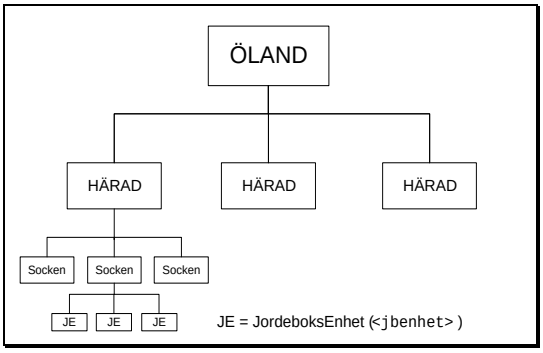
Den base märkordsuppsättning som används här är TEI.prose, den enklaste av de base-uppsättningar som finns i TEI. De utökade märkordsuppsättningar som kommer till användning är:

- TEI.figures, element för bilder och illustrationer av diverse format
- TEI.nets, element för bl.a. släkträd och tabeller
- TEI.pointers, element för externa referenser mellan olika dokument

Jordeboksmaterialets struktur

Det jordeboksmaterial som ska SGML-märkas är hämtat ur den på papper publicerade volymen Öland, utgiven av Riksantikvarieämbetet i serien ”Det medeltida Sverige” nr 4/3, 1996. Denna publicerade volym inleds med ett allmänt kapitel om Öland, som innehåller information såsom topografiska huvuddrag, administrativ indelning o.s.v. – fakta som rör hela Öland. Detta inledande kapitel består av vanlig brödtext, rubriker samt några andra element, det är en relativt enkel struktur och kodas i huvudsak

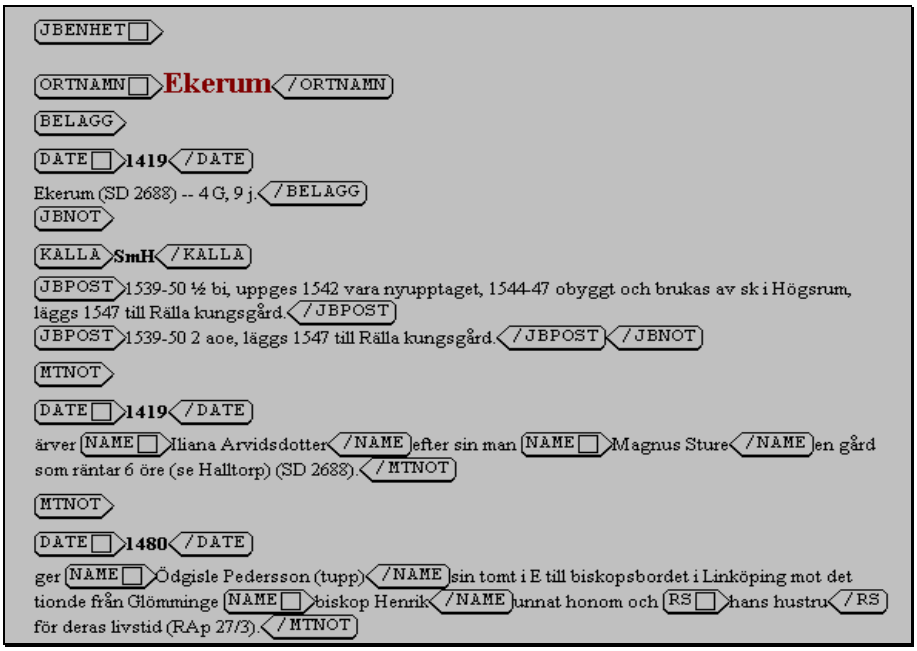
med TEIs styckeelement <p> och rubrikelement <head>. Sedan följer beskrivningen av olika härader som Öland är uppdelat i.



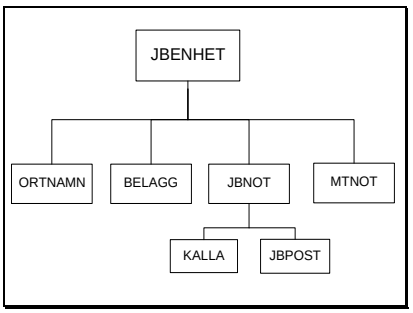
Figur 3 Jordeböckernas struktur

Ur ovanstående figur kan utläsas att Öland består av flera härader, ett härad består av socknar, en socken består av jordeboksenheter (oftast by/gård, kan även vara kvarn el. äng). Härad och socken motsvaras av div-element. Genom att ge div-elementets attribut `type` ett lämpligt värde, kan härader och socknar skiljas åt (`<div1 type='harad'>, <div2 type='socken'>`). För jordeboksenhet konstruerades ett eget chunkelement, `<jbenhet>`. Detta element har ett attribut `type` som kan anta tre olika värden: `gard` (gård), `kvarn` eller `ang` (äng).

Partierna som motsvarar elementen härad och socken är relativt okomplicerade, de består av en inledning följt av information om socknar resp. byar. Däremot är elementet `<jbenhet>` något mer komplicerat uppbyggt, en märkning av `<jbenhet>` ses i nedanstående figur.



Figur 4. Märkning av elementet `<jbenhet>`



Figur 5. Element Jordeboksenhet

Bilden här till vänster visar strukturen hos elementet `<jbenhet>`. Först kommer ett ortnamn som talar om vilken jordeboksenhet som redovisningen gäller, följt av elementet `<belagg>` (årtalet första gången ett element har nämnts i de källor som används). I belagg finns elementen `<datum>`, `<ortnamn>`, `<namnform>`, och `<kalla>` i angiven ordning.

Elementet `<jbnot>`, jordeboksnotis, innehåller information från Smålands handlingar[1] och bygger därmed på inventering av 1500-talets kamerala material. `Jbnot` innehåller en eller flera `<jbpost>`, jordebokspost. En jordebokspost innehåller information om jordens brukning, jordnatur (typ av jord och ägare), och årtal.

Efter detta följer ett eller flera element av typen `<mtnot>`, medeltidsnotis. Det beskriver de förändringar som har skett vad gäller ägandeförhållanden av marken. Förändringar kan vara resultatet av arv, köp eller gåva. Element som antas vara intressanta att märka ut här är framför allt namn. Genom att märka ut alla namn kan man kartlägga godsägare under medeltiden, var de ägde jord någonstans, hur mycket jord de ägde med mera.

I den tryckta volymens sista kapitel finns en lista över förkortningar och källor som är använda i jordeboken. Dessa är uppmärkta med TEIs element för att märka upp listor, elementen är beskrivna i TEI guidelines [18] kap 6.7. De element som här använts är `<list>`, `<item>` och `<label>`.

Modifieringar av TEIs DTD

Jordeböckerna innehåller en hel del information som det var svårt att finna några passande element för i TEI. En tänkbar lösning vore att koda denna information med TEIs `phrase-` eller `chunk-`element och sedan tilldela elementets attribut ett värde som mer explicit talade om vilken typ av information som finns i elementet. Men då många av dessa element beskriver jordeböckernas struktur så är det lämpligare att använda element framför attribut. De element som lagts till är:

- `kartkod` beskriver kartkod för en `<jbenhet>`
- `ortnamn` namn på ort
- `jbenhet` jordeboksenhet kan vara: by, gård, kvarn eller äng
- `jbnot` jordeboksnotis, innehåller källa, följt av jbpost,
- `jbpost` jordebokspost, typ av jord, storlek och ägare
- `mtnot` medeltidsnotis, förändringar i jordenaturen
- `belagg` belägg för första gång en bys eller sockens namn förekommer i källmaterialet
- `kalla` källhänvisning
- `namnform` äldre namnformer på t.ex. ortnamn

[1] Sammanfattande benämning på de i Kammarkivet förvarade landskapshandlingarna från ca. 1540–1560.

För en detaljerad beskrivning av de ändringar som gjorts av TEIs DTD, se filerna `oland.ent` och `oland.dtd` som återfinns i bilaga 1.

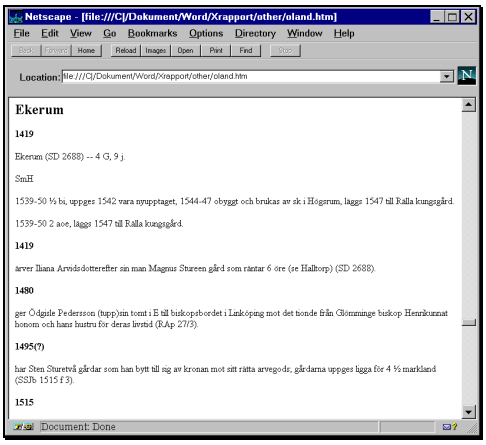
Arbetsgång vid märkning och konvertering

Jordeboksmaterialet levererades från Riksantikvarieämbetet som filer i Wordformat. Dessa filer har sparats som ren ASCII-text. Det märkningsarbete som sedan följer är relativt manuellt. Editorn Emacs har använts för att SGML-märka texten, det går relativt fort om man är van. Märkorden sätts in ”för hand” med hjälp av kortkommandon som är definierade i Emacs SGML-mode. Uppskattningsvis tar det ca. 3–4 veckor att märka hela Ölandsvolymen (340 sidor), jag har dock endast märkt ut ca. 25 sidor för att illustrera tekniken. De sidor som är uppmärkta är 97–119, 139–144, 327 och 337. Sid 113–119 av dessa sidor är noggrant uppmärkta.

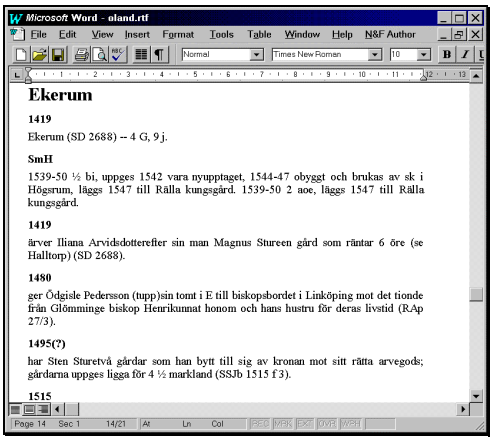
Som tidigare nämnts finns det konverteringsprogram som konverterar från RTF-format till en speciell DTD kallad Rainbow-DTD, som eventuellt kan konverteras till TEIs DTD med hjälp av Perl-paketet Sgmlspm. Denna procedur har inte studerats närmare, men om det blir aktuellt att märka ut flera böcker kan det vara värt att studera detta mer ingående för att automatisera märkningsarbetetet, och på så sätt spara tid.

Konvertering till RTF och HTML

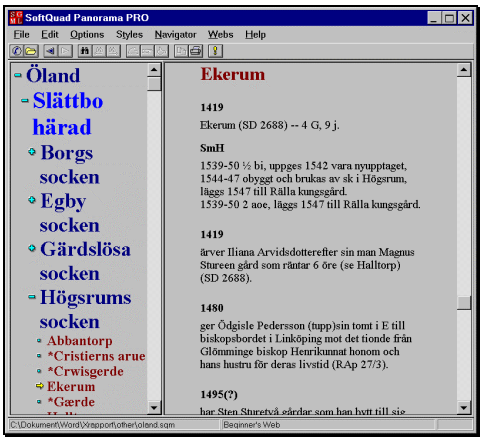
Med hjälp av konverteringsprogrammen beskrivna i föregående kapitel har det SGML-märkta jordeboksmaterialet konverterats till RTF och HTML. För att konvertera till HTML har Perl-paketet Sgmlspm använts. Och med hjälp av ett TEI DSSSL-stylesheet har JB konverterats till RTF-format.



Figur 6. HTML-version av JB



Figur 7. RTF-version av JB inläst i MS Word



Figur 8. SGML-version av JB

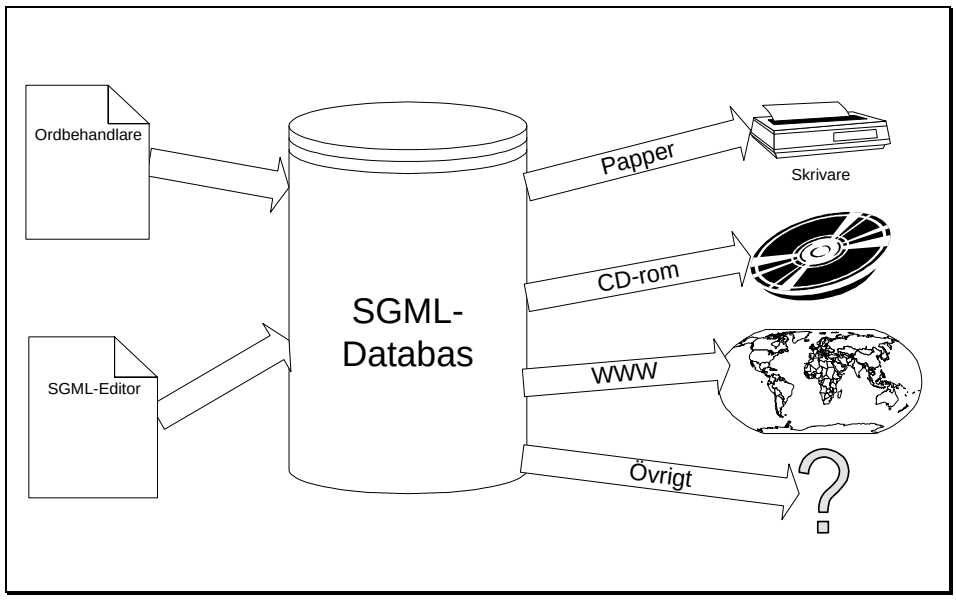
Användarsituation och Protoyp

Tänkbar användarsituation

Med en väl utförd SGML-märkning kan bl.a. dessa två användargrupper ha glädje av jordeboksmaterialet i elektronisk form:

- professionella användare som nyttjar materialet i sitt yrke, exempelvis historieforskare, ortnamnsforskare. Dessa har höga krav på sökfunktioner och vill ofta ha svar på avancerade frågeställningar.
- amatöranvändare, t.ex. släktforskare och skolelever. Dessa kan nöja sig med en HTML-version av jordeboksmaterialet och har inte samma krav på sökmöjligheter. Här är det däremot viktigare att dokumentet anpassas till läsarens förutsättningar och behov. Åtgärder för att åstadkomma detta kan vara att expandera förkortningar som nu finns i TEI-versionen.

Nedanstående bild är en grov skiss över ett tänkbart system för en SGML-databas. På vänster sida syns indata till databasen, och på höger sida syns utdata.



Figur 9 SGML-Databas, tänkbart system

Uppmärkningen av ett TEI-dokument kan ske antingen i en SGML-Editor eller i en vanlig ordbehandlare. I det senare fallet krävs någon form av konverteringsfilter innan dokumentet läggs in i databasen. Som utdata kan man tänka sig följande:

- utskrift på papper

- utmatning på CD-Romskiva
- koppling till WWW, antingen via någon HTML-läsare eller genom att användaren förses med specialprogram för att logga in till databasen via telnet. Prototypen som beskrivs nedan är ett förslag på ett enkelt användargränssnitt för ett sådant program.
- frågetecknet står här för andra möjliga användningsområden, t.ex. koppling till GIS (Geografiska Informations System), som är en standard för geografisk information rörande bl.a. kartmaterial.

Prototypen

Prototypen som beskrivs i detta avsnitt, kan ses som ett gränssnitt till en databas med TEI-kodad litteratur. Den riktar sig i första hand till professionella användare och ska underlätta för denne att hålla ordning på vad som finns i en databas. Men den kan självklart även användas av amatöranvändare. Programmet är en kombinerad TeiHeader-läsare och enkel SGML-läsare. Ett krav på programmet var att det skulle gå att söka i flera olika TEI-dokument samtidigt. Prototypen innehåller än så länge föga funktionalitet, utan avser att demonstera principer för ett tänkbart gränssnitt.

Windows programutveckling val av verktyg

Givet från början var att prototypen skulle utvecklas i Windows. Att skriva ett minimalt Windowsprogram som öppnar ett fönster och skriver ut ”hello” tar ca. 50–100 rader kod. Så det är en relativt hög tröskel att komma igång med windowsprogrammering.

Det verktyg jag valt att använda är Microsoft Visual C++. Det finns diverse olika visuella programspråk som underlättar skapandet av grafiska gränssnitt i Windowsmiljö. Visual Works Smalltalk, Borland C++ och Visual C++ är de jag testat eller haft erfarenhet av. Valet föll till slut på Visual C++, bl.a. beroende på följande orsaker:

- det är relativt enkelt att skapa gränssnitt med dialogboxar och menyer
- behaglig programmeringsomgivning att jobba i
- en prototyp som är skriven i Visual C++ går att utveckla vidare utan att man behöver börja om från början, till skillnad från en del andra ”visual”-språk.

Visual C++ har ett ”ramverk”, MFC (Microsoft Foundation Classes). Det fungerar som ett gränssnitt mellan Windows API (Application Programming Interface) och användaren. Detta ramverk underlättar windowsprogrammering, och man behöver inte lära sig alla de 700 funktioner som finns i Windows API. MFC har använts vid utveckling av denna prototyp.

Synex ViewPort, en SGML-motor

Den prototyp som tagits fram är gjord med hjälp av Synex ViewPort (VP), som tillhandahåller en mängd funktioner och datastrukturer för att tillföra SGML-kompatibilitet till befintliga program. VP tar hand om alla detaljer som rör SGML-arbetet såsom att kontrollera att ett SGML-dokument är giltigt, lösa

entiteter för publika identifierare, visa dokumentet enligt tilhörande stylesheet, visa tillhörande navigator med mera. Arbetsgången är något förenklat den, att man först konstruerar sitt gränssnitt, och sedan anropar de funktioner som finns tillgängliga i VP. Alla utvecklingsverktyg/programspråk som klarar av att anropa C-funktioner kan användas. Det finns ca. 250 C-funktioner man kan anropa i VP. Nedan beskrives de vanligaste funktionerna:

- `SvOpenNameDoc` öppnar ett SGML-dokument
- `SvOpenDefaultNav` öppnar en navigator till ett SGML-dokument
- `SvDisplayPage` visar ett SGML-dokument eller navigator i ett fönster

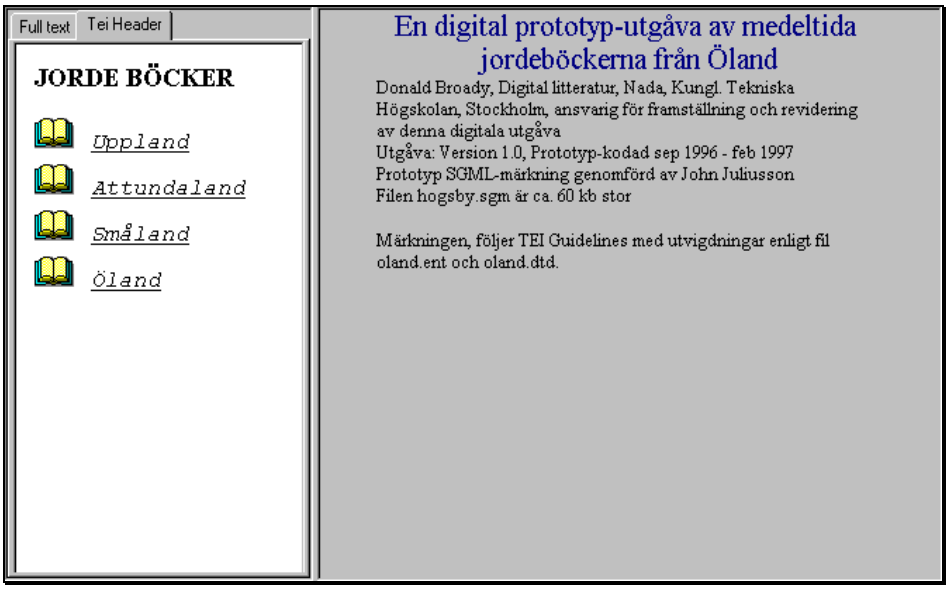
Med hjälp av VP kan man dessutom förse sitt program med Hytimelänkar[1], grafik, bilder, video och ljud. ViewPort är utvecklat i förlängningen av en forskning som inleddes vid Nada, KTH och säljs i dag av Synex som är ett fristående företag. ViewPort-teknologin ligger också bakom den första SGML-browsern, Panorama.

Prototypen, guidad tur.

Varje TEI-dokument är försedd med ett element `<TeiHeader>`, som innehåller s.k. metainformation om dokumentet: författare, ISBN-nr för en ev. tryckt förlaga, titel, sökord med mera. Den innehåller även speciell information om den digitala utgåvan: storlek i MB, ansvarig för digital utgåva med mera. Denna TeiHeader kan jämföras med de kartotekskort som finns på ett bibliotek. På ett liknande sätt ska man kunna läsa dessa TeiHeaders.

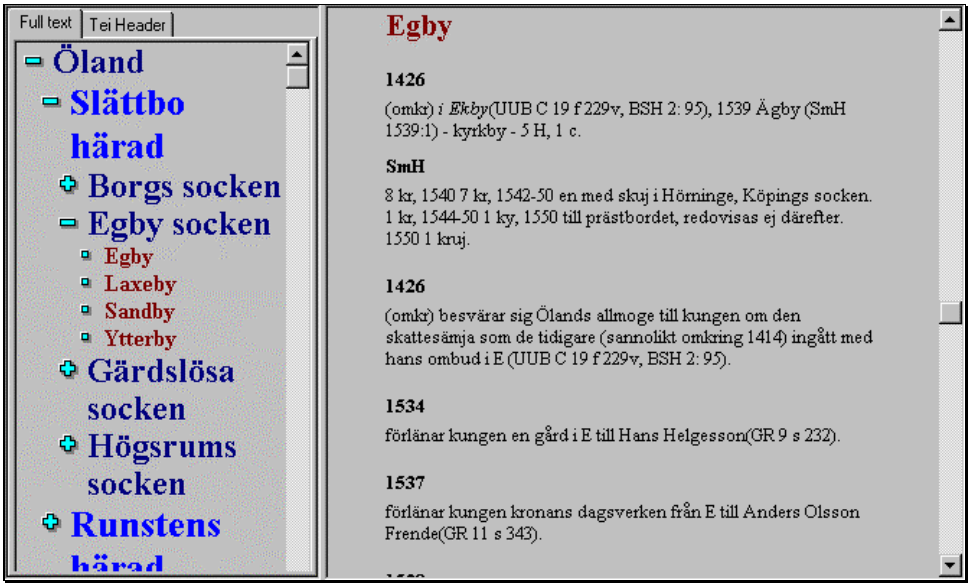
Funktionaliteten i detta prototypprogram är relativt enkel. På vänster del av skärmen syns ett fönster där man kan växla mellan två lägen, TeiHeader och Fulltext. Nedan syns en skärmdump från TeiHeader-läget. På vänstra delen av skärmen syns här de filer med jordeboksmaterial som finns i databasen. När man klickar på någon av dessa så visas dess TeiHeader i det högra fönstret.

[1] Hytime är en internationell standard (ISO 10744:1992) för att hantera dels hypertextlänkar, dels tidsberoende information (ljud, rörliga bilder).



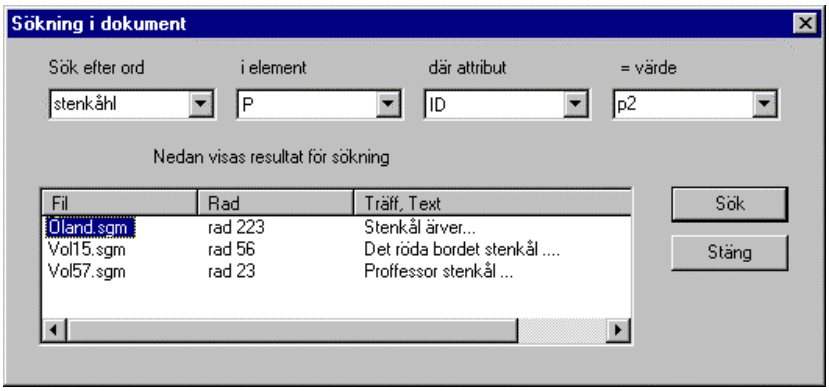
Figur 10. Prototyp i TeiHeader-läge

För att visa hela dokumentet byter man till Fulltext-läget. Detta görs genom att klicka på Fulltextfliken. Då syns navigatorn i vänstra fönstret, och hela dokumentet syns i högra fönstret. Det dokument man får upp är det som var aktivt i TeiHeader-läget. Här kan man bläddra och läsa dokumentet som i en vanlig SGML-läsare.



Figur 11. Prototyp i Fulltext-läge

För att söka i en dokumentsamling, välj ”Sök->I dokument-samling” från menyn. Fyll i det ord du söker efter. Genom att välja ett specifikt element med attribut i comboboxorna finns här möjlighet att göra avancerade sökningar. Sökresultatet presen-teras i listan längst ner i dialogrutan. När man dubbelklickar på någon av filerna i listan under ”Fil” visas filen i fråga i det högra huvudfönstret



Figur 12. Sökdialog

Resultat, slutsatser

Här presenteras resultat och praktiska erfarenheter av detta examensarbete samt några förslag till framtida studier. Slutsatser från tidigare avsnitt i rapporten kan komma att upprepas.

Resutat och erfarenheter

Verktyg för arbete med SGML och TEI

Ett led i arbetet var att utpröva lämpliga datorverktyg, framför allt editorer för SGML-märkning samt för konstruktion och editering av DTD:er. Eftersom SGML och TEI representerar en tämligen ny teknologi saknas det än så länge bra lättanvända programvaror.

Vi valde editorn Emacs med tillägget SGML-mode i kombination med SGML-validatorn Nsgmls, vilket för våra syften fungerade utmärkt för SGML-märkning av jordeboksmaterialet samt vid DTD-konstruktionen. Emacs är en kraftfull editor som klarar stora textmängder och vars sök och ersätt-funktioner etc. var till stor hjälp. För konvertering mellan olika format (SGML, HTML, RTF) användes skript skrivna i programspråket Perl.

TEIs DTD är en komplex tillämpning av SGML. Den innehåller många element (totalt 400) och är avsedd för många skilda användningsområden inom humaniora. Det stora arbetet har här varit att sätta sig in i hur denna DTD är uppbyggd. Goda kunskaper om SGML i allmänhet och om TEIs DTD i synnerhet krävs när man behöver göra egna tillägg till DTD:n, vilket jordeboksmaterialet fordrade. När man väl besitter sådana kunskaper går det dock snabbt att revidera DTD:n. Oftast krävs inte särskilt mycket nyskriven SGML-kod.

Denna arbetsmiljö byggd kring editorn Emacs är dock inte att rekommendera för användare utan programmeringskompetens. Det existerar ett antal mer användarvänliga editorer med grafiskt användargränssnitt, men dessa tillåter inte ett lika snabbt och effektivt arbete. Vanligt är att man inte kan utnyttja TEIs modulära uppbyggnad utan tvingas ”normalisera” TEIs DTD innan editorn kan användas.

När väl DTD:n är utvecklad kan en mer lättanvänd editor duga. Då är det inte mycket svårare att SGML-märka materialet än att använda ett vanligt ordbehandlingsprogram. Den person som ska genomföra SGML-märkningen behöver inte ha några djupare kunskap om SGML. Dock behövs vissa insikter i DTD:ns uppbyggnad, så att man kan bedöma hur och var de olika märkorden får placeras.

Dokumentanalys, anpassning av TEIs DTD

De existerande tryckta utgåvorna av jordeboksmaterialet är informationsrika och har en mycket komplex struktur. För att bevara denna informationen i en digital utgåva krävs att man förstår hur de tryckta böckerna är uppbyggda, vad förkortningarna innebär etc. Denna dokumentanalys är en ganska tidskrävande uppgift.

TEIs DTD har visat sig vara lämplig. Tack vare den modulära uppbyggnaden är den flexibel och lätt att modifiera. Man kan utan svårighet lägga till egna element och attribut. På så sätt har jordeboksmaterialets struktur och innehåll kunnat representeras i den digitala versionen. Om syftet enbart varit att göra materialet presenterbart på datorskärm skulle en HTML-märkning ha kunnat duga, men mycken information hade då gått förlorad.

Utveckling av prototyp

Den prototyp jag tagit fram illustrerar hur ett användargränssnitt för en samling TEI-kodad information kan te sig, exempelvis i en framtid hela jordeboksmaterialet.

Gränssnittet för denna prototyp utnyttjar den information som finns i den del av TEI-dokumenten som kallas Tei Header, dvs. metainformation om dokumentets innehåll. För att förse prototypen med SGML-funktionalitet användes SGML-motorn Synex Viewport, vilket var tämligen enkelt. Den största hindret var att tröskeln är ganska hög innan man lär sig programmera i Windows-miljö.

Framtida studier

Databaslösningar

Prototypen tar inte hänsyn till hur filerna är lagrade. De skulle kunna lagras som textfiler i ett bibliotek eller i en databas. I framtiden torde ofta någon form av databaslösning vara att föredra i de fall då materialet är stort och komplext. På grund av att WWW nått en sådan spridning kommer troligtvis SGML-databaser med WWW-gränssnitt att bli allt vanligare. I dag finns det några konkurrerande teknologier (relations- eller objekt-databasteknologier eller utvecklingar därav) för design av SGML-databaser.

En jämförande studie av dessa teknologier, med inriktning mot hur framtida TEI-databaser kan designas och indexeras, vore värdefull.

DSSSL och XML, hur ser framtiden ut?

De två nya standarderna XML och DSSSL kommer troligtvis att få betydelse för elektronisk publicering. XML kan i vissa sammanhang komma att ersätta HTML som märkspråk för WWW. Många tongivande dataföretag (Microsoft, Sun) har ställt sig positiva till XML. När denna standard är klar dröjer det troligtvis inte så länge förrän XML-läsare blir tillgängliga via Internet, då tekniken redan existerar i dagens SGML-läsare. DSSSL har än så länge

fått en trög start, det finns få programvaror som stödjer denna standard.

En undersökning av hur dessa två standarder kommer påverka SGML-tillämpningar i framtiden vore av intresse för fortsatt arbete i den riktning som inletts med detta examensarbete.

Litteraturförteckning

Windowsprogrammering

[1] Eriksson, Hans Erik (1994): *Programutveckling för Windows och Windows NT*. Lund: Studentlitteratur.

[2] Broquard, Vic (1996): *Programming with MFC for Windows 95*. Upper Sadle River: Prentice Hall.

[3] [Diskussionsinlägg i newsgroup] *comp.os.ms-windows.programmer.tools.mfc*.

SGML

[4] Herwinjen, Eric Van (1994): *Practical SGML Second edition*. London: Kluwer Academic Publishers.

[5] Travis, Brian/Waldt, E./ Dale C. (1995): *SGML Implementation Guide*. Berlin: Springer-Verlag.

[6] [Diskussionsinlägg i newsgroup] *comp.text.sgml*.

[7] [Dokumentation till datorprogrammet James Clark Sgml Parser kit, Nsgmls] `http://www.jclark.com/sp`

[8] Smith ,Joan M. (1992): *SGML and Related standards document description and processing languages*. New York: Ellis Hoorwood.

Perl

[9] Quicgely, Ellie (1995): *Perl by example*. Upper Sadle River: Prentice hall PTR.

[10] [Dokumentation till konverteringskit SGMLS.pm (A perl5 class library for handling output from the NSGMLS parsers)] URL `http://www.uottawa.ca/~dmeggins/SGMLSpm/sgmlspm.html`.

[11] *Perl5 Manual*, URL `http://www.metronet.com/perlinfo/perl5.html`.

DSSSL, XML

[12] *XML Working Draft 14-Nov-96*, URL `http://www.textuality.com/sgml-erb/WD-xml-lang.html`.

[13] *XML faq*, URL `http://www.ucc.ie/xml/`.

[14] *DSSSL Final Draft*, URL `http://occam.sjf.novell.com:8080/dsssl/dsssl96/1.toc`,

[15] [Dokumentation till James Clark's DSSL-program Jade] URL `http://www.jclark.com/jade/`

Emacs

[16] Cameron, Debra/Rosenblatt, Bill (1992): *Learning GNU Emacs*. Sebastapol: O'Reilly & Associates.

[17] *Elisp manual*, `http://www.cs.indiana.edu/usr/local/www/elisp/lispref/elisp_toc.html`.

Digital Litteratur

[18] *TEI-P3, Guidelines for Electronic Encoding and Interchange*. Chicago/Oxford: Text Encoding Initiative, URL `http://etext.virginia.edu/TEI.html`.

[19] Broady, Donald (1993): *Det nya handbiblioteket*. Rapport IPLab-73. Stockholm: Nada, KTH.

[20] Ide, Nancy/ Véronis, Jean (1995): *Text Encoding Initiative. Background and Context*. Dordrecht: Kluwer Academic Publishers.

[21] American Verse Project, URL
`http://www.hti.umich.edu/english/amverse/texts/index.html`

[22] The University of Virginia Electronic Text Library, URL
`http://www.lib.virginia.edu/etext/ETC.html`

Jordeboksmaterialet

[23] Axelsson, Roger/Janzon, Kaj/Rahmqvist, Sigurd (utg.) (1996). *Öland*. Det Medeltida Sverige, vol 4:3. Stockholm: Riksantikvarieämbetet.

[24] Ramqvist, Sigurd/Skoglund, Lars-Olof (utg.) (1986): *Uppland.* Det Medeltida Sverige, vol 1:5. Stockholm: Riksantikvarieämbetet.

Samtliga HTTP-adresser i denna referenslista är kontrollerade 970503.

Bilaga 1

Egna elementdefinitioner till TEIs DTD.

TEI.extensions.ent, filen Oland.ent

```
<!-- I denna fil görs ändringar av entiteter eller läsas in entiteter -->
<!-- Teckenuppsättning latin1  -->
<!ENTITY % ISOlat1 public "ISO 8879:1986//ENTITIES Added Latin 1//EN">
%ISOlat1;


<!-- genom att lägga till element efter x.??? bestäms var element får
förekomma -->


<!-- omdefiniering av elementet p -->
<!ENTITY % p 'IGNORE'                                        >


<!-- nya element i m.data -->
<!ENTITY % x.data 'ortnamn | kartkod | jbpost | kalla| namnform |'>


<!-- nya element i m.chunk -->
<!ENTITY % x.chunk 'jbenhet | jbnot | mtnot | belagg | '  >


<!-- nya element i m.common -->
<!ENTITY % x.common 'table |'                                >
```

TEI.extensions.dtd, filen Oland.dtd

```
<!-- I denna fil deklareras nya element och deras attribut -->


<!-- element på frasnivå -->
<!-- Ortnamn -->
<!ELEMENT ORTNAMN - - (%phrase;)                             >
<!ATTLIST ORTNAMN
                %a.global;
                %a.names;
                type CDATA #IMPLIED                          >


<!-- Kartkod -->
<!ELEMENT KARTKOD - - (%phrase;)                             >


<!-- Källhänvisning -->
```

```
<!ELEMENT KALLA - - (%phrase;)                              >


<!-- element på paragrafnivå -->


<!-- Jordeboksenhet (by) -->
<!ELEMENT JBENHET - - (ortnamn, ((belagg*, jbnot*, p?, mtnot*) | ref))
+(%n.graph;)                                                >
<!ATTLIST JBENHET type (gard|kvarn|ang) #IMPLIED            >


<!-- Jordeboksnotis -->
<!ELEMENT JBNOT  - - (%phrase.seq;)                         >


<!-- Jordebokspost -->
<!ELEMENT jbpost  - - (%phrase.seq;)                        >


<!-- Medeltidsnotis -->
<!ELEMENT mtnot  - - (%phrase.seq;)                         >


<!-- Belägg - första gång ett ort/socken-namn har nämnts -->
<!ELEMENT belagg - - (%phrase.seq;)                         >


<!-- namnform, t.ex äldre ortnamn -->
<!ELEMENT namnform  - - (%phrase.seq;)                      >


<!-- Omdefiniering av elementet p -->
<!ELEMENT p - o (%paraContent;)                             >
<!ATTLIST p %a.global;
      type (omf|kb|rs)  #IMPLIED                            >
```