Latvijas Universitāte
Fizikas un matemātikas fakultāte
Datorikas nodaļa

LATVIEŠU LITERATŪRAS KORPUSA PROGRAMMRĪKU IZSTRĀDE

kursa darbs

Autors
Normunds Grūzītis (DatZ990026)

Vadītājs
Kalvis Apsītis,
Dr. Sci. Comp.,
Latvijas Universitātes docents

Rīga, 2002

ANOTĀCIJA

Šajā darbā ir aplūkota latviešu literatūras korpusa programmrīku projekta izstrādes gaita, kā arī XML un Interneta programmēšanas iespējas valodas korpusa veidošanā. Nozīmīgākā daļa ir atvēlēta sistēmas projektējumam un pirmajam izstrādes posmam – esošā mašīnlasāmā tekstu masīva automātiskas strukturēšanas iespējām kvalitatīvā, saturu aprakstošā marķējumā. Apskatītas ir arī iespējas un plānotie risinājumi XML bāzētu tekstu apstrādē literatūras korpusa mērķauditorijai.

ABSTRACT

In this work a process of software development for Latvian literature corpus and facilities of XML and web programming in language corpus development are covered. Major part of this work is dedicated to project design and to the first stage of development – facilities of automatic structuring of existing machine-readable text massive into qualitative, content describing mark-up. Facilities and planned solutions in processing of XML-based texts for literature corpus end-users are also overviewed.

АНОТАЦИЯ

В этой работе разсматривается ход разработки проекта программы пользователя корпуса латышской литературы и возможности технологий XML и программирований Интернета в создании языкого корпуса. Важнейшая часть посвещена проектирование системы и первой ступени разработки – возможности автоматической структуризаций текстовых массивов в качественной маркировке. В работе также рассмотренны возможные и намечаемые решения обработки текстов на базе XML конечному пользователю корпуса.

SATURA RĀDĪTĀJS

IEVADS

Šajā darbā ir aplūkota latviešu literatūras korpusa programmrīku projekta izstrādes gaita un XML un Interneta programmēšanas iespēju izmantošana valodas korpusa veidošanā. Projektējuma izstrāde un sistēmas pirmās kārtas realizācija ar Kultūrkapitāla fonda atbalstu notiek Latvijas Universitātes (LU) Matemātikas un informātikas institūtā (MII), un tajā tehnisko risinājumu izpētē un izstrādē sadarbībā ar valodniekiem un korpuslingvistikas speciālistiem piedalās darba autors.

Aplūkotā problemātika un risinājumi var noderēt ne tikai valodas korpusu veidošanā, bet arī portālu izstrādē vai citā sfērā, saskaroties ar dažādām tekstu uzkrāšanas, apstrādes un analīzes problēmām.

Valodas korpuss

Valodas korpuss ir liels, mašīnlasāmu (*machine-readable*) tekstu masīvs. Balansēts valodas korpuss ir korpuss, kas maksimāli reprezentē šīs valodas funkcionālo stilu daudzveidību [McEnery, Spektors]. Literatūras korpuss ir viena no valodas korpusa apakškopām, un tas aptver visas valodas korpusa funkcionālās īpašības un prasības.

Valodas korpusu praktiskā lietojamība ir plaša – tos izmanto vārdnīcu veidošanā, terminoloģijas ekstraģēšanā, tekstu tulkošanā, valodas apmācībā (gūst lielu popularitāti), lingvistisko programmatūru izstrādē, apmācībā un testēšanā (piem., mašīntulkošana, tekstu skenēšanas un atpazīšanas – OCR – programmas), valodniecības pētījumos, izglītībā, kā arī citās sfērās.

Funkcionāls darbs ar korpusu, protams, nav iedomājams bez atbilstošu programmrīku kopuma. Lai nodrošinātu pilnvērtīgu tekstu analīzi, korpusa saturu nepieciešams marķēt. Ir vairāku veidu marķēšanas klasifikācijas, bet datorlingvistikā ir vispārpieņemti pieci tekstu marķēšanas līmeņi [Spektors]:

0. līmenis: nemarķēts teksts, satur tikai dabīgās iezīmes: pieturzīmes, rindas pārnesumus utml.
1. līmenis: strukturālā marķēšana – apraksta teksta saturu, piemēram, norāda katras rindkopas vai panta struktūru, sākumu un beigas;

2. līmenis: morfosintaktiskā marķēšana – katram vārdam norāda tā gramatisko raksturojumu;
3. līmenis: sintaktiskā marķēšana – teikuma struktūras un teikuma locekļu identifikācija;
4. līmenis: semantiskā marķēšana – aiz katra vārda (kā arī idiomām un frazeoloģismiem) norāda visus tā semantiskos raksturojumus.

Esošā situācija

Ar valodas korpusu veidošanu – tekstu uzkrāšanu un programmnodrošinājuma izstrādāšanu – nodarbojas organizācijas daudzās valstīs, piemēram, Birmingemas Universitāte (*Bank of English*), Oksfordas Universitāte (*British National Corpus*), Pensilvānijas Universitāte (*Penn Treebank*), Vācu valodas institūts (*Textkorpora*), Tartu Universitāte (igauņu rakstītās valodas korpuss) u.c. Latvijā par balansēta latviešu valodas, tajā skaitā literatūras, korpusa izveidošanu neviena institūcija vēl nav paziņojusi, bet eksistē vairāki Interneta portāli, kuros pieejami apjomīgi tekstu masīvi, kas reprezentē nozīmīgus valodas stilus. Piemēram, ziņu un laikrakstu datubāzes (BNS, *LETA*, *Lursoft*, *Diena*, *DELFI*, u.c.), juridisko dokumentu datubāzes (NAIS) un citas līdzīgu kategoriju informatīvās sistēmas, tiesa gan, šo sistēmu izstrādes mērķis nav bijis valodas korpusa funkcionalitātes nodrošināšana. Tomēr interese un nepieciešamība pēc dažādu funkcionālo stilu pārstāvēta latviešu valodas korpusa arvien palielinās.

Līdz šim LU MII ir daudz paveikts latviešu valodas korpusa uzkrāšanā [Milčonoka], izmantojot OCR tehnoloģijas un manuālu tekstu ievadi, kopā ir uzkrāti aptuveni 20 milj. vārdlietojumu (salīdzinājumam, *Bank of English* apjoms ir 320 milj. vārdlietojumu), no kuriem literatūras klasika aptver aptuveni 3.5 milj. [`www.ailab.lv/Teksti`]. Tomēr pilnvērtīgam korpusam ir nepieciešama ne tikai apjomīga tekstu kolekcija, bet arī vienota koncepcija, programmrīki un darba vide, kas nodrošinātu:

1. tekstu glabāšanu standartizētā, marķētā (kādā no iepriekšminētajiem līmeņiem) un apstrādāšanai ērtā formātā;
2. navigācijas un meklēšanas iespējas visā korpusa saturā;
3. korpusa specifiskās iespējas: vārdlietojumu biežuma analīzi, vārdu konkordances (apkaimes), vārdu savienojumu analīzi u.c.;

4. interfeisu ar korpusu un tā funkcijām (Interneta vidē) gan profesionālajiem lietotājiem, gan pārējiem interesentiem.

Pirmais punkts – glabāšanas formāts, struktūra un datu modelis – ir ļoti svarīgs visa korpusa veidošanas procesā. Tas ir pamats, uz kura balstās un no kura ir atkarīga visu pārējo korpusa izstrādes posmu veiksmīga realizēšana un arī vēlāko izmaiņu un uzlabojumu sarežģītība un elastība.

Pašlaik LU MII uzkrātais valodas korpuss (teksti, kas saskaņā ar autortiesību likumu ir brīvi publicējami) ir veidots ar mērķi to padarīt plaši pieejamu, publicējot Internetā. Visi teksti ir HTML marķēti un principā var uzskatīt, ka esošais korpuss atbilst tekstu marķēšanas pirmajam līmenim, jo tas noteikti ir solis tālāk, salīdzinot ar pilnīgi nemarķētu tekstu (ja neskaita dabiskās marķējuma iezīmes - pieturzīmes). Tomēr šādi korpuss nav marķēts strukturāli - saturu aprakstoši, bet tikai vizuāli, aprakstot tekstu vizuālo struktūru un atainošanu. Turklāt nav bijusi izstrādāta vienota koncepcija un tekstu marķēšanas standarti, kas rada dažādas problēmas un grūtības automātiskā korpusa satura atpazīšanā un strukturēšanā, lai iegūtu kvalitatīvi marķētu korpusu.

Šobrīd notiek darbs pie literatūras korpusa programmrīku projekta pirmās kārtas projektējuma izstrādāšanas, un praktiskā realizācija pašlaik atrodas sākumstadijā: standartizētas 1. līmeņa marķēšanas koncepcijas izstrādē un atbilstošu automātisku satura atpazīšanas un strukturēšanas rīku izveidē. Līdz ar to šīs būs galvenās darbā aplūkotā tēmas. Bet darba turpinājumā tiks apskatītas iespējas, kādas paveras uz XML un *Java* tehnoloģijām bāzētu rīku izstrādē korpusa galalietotājiem.

Projekta mērķis

Projekta pirmās kārtas mērķis ir nodrošināt lielāko mērķauditoriju lietotājus ar būtiskākajām korpusa funkcionālajām iespējām Interneta vidē. Uzdevums ir izstrādāt sistēmas projektējumu un korpusa tekstu aprakstīšanas un glabāšanas standartus, kā arī metodiku, kas nodrošinātu stabilu un elastīgu pamatu tālākai praktiskai realizācijai gan izstrādes šajā kārtā, gan nākotnē, paplašinot sistēmu. Pirmais posms ir programmrīku izstrādāšana visu esošo literatūras korpusa tekstu iespējami automātiskai transformēšanai standartizētā formātā.

I KORPUSA IZSTRĀDES PRASĪBAS

Visas tekstu strukturēšanas standartu, datu modeļu un funkcionālās prasības sistēmas izstrādei tika noteiktas un definētas, sadarbojoties ar LU MII korpuslingvistikas speciālistiem. Analizējot šīs prasības, ir izstrādāts sistēmas projektējums (skatīt nākošo nodaļu), uz kura pamata pašlaik notiek sistēmas realizācija.

1.1. Korpusa lietotāju kategorijas

Korpusa potenciālie lietotāji aptver plašu auditoriju, un tos var iedalīt trīs galvenajās kategorijās:

- **akadēmiskie lietotāji** – valodnieki, leksikogrāfi, literāti, datorlingvisti, korpuslingvisti un citi, kas korpusu izmantos zinātniski pētnieciskiem, izglītojošiem un nekomerciāliem mērķiem;
- **komerciālie lietotāji** – pārsvarā institūcijas, kas korpusa saturu izmantos komerciāliem mērķiem, piemēram, izdevniecības vārdnīcu sastādīšanai un literāro darbu izdošanai (saskaņā ar autortiesībām), IT firmas, kas nodarbojas ar valodas apstrādi, programmu testēšanai, apmācībai un datu bāzēm;
- **interesenti** – pārējie lietotāji, kurus interesēs tikai pats korpusa saturs – literārie teksti, nevis uz šiem tekstiem balstītu pētījumu veikšana (piem., skolnieki, kas vēlas izlasīt kāda autora darbu vai uzzināt kāda darba autoru, sameklēt kāda zināma citāta avotu utml.).

Ārpus iepriekšminētajām atsevišķu kategoriju veidos korpusa **administratīvie** lietotāji, kas nodarbosies ar korpusa satura papildināšanu un kvalitātes uzlabošanu – metadatu un saturisku kļūdu labošanu, augstāka līmeņa marķēšanu, programmrīku izstrādāšanu un sistēmas uzturēšanu.

Attiecībā uz korpusa auditorijas sadalījumu lietotāju skaita ziņā, paredzams, ka lielāko daļu aizņems akadēmisko lietotāju kategorija un interesenti, bet ievērojami mazāk lietotāju būs no komerciālā sektora. Tomēr, ar laiku palielinot korpusa saturu un

tā pievienoto vērtību, kā arī attīstot programmrīku funkcionalitāti, pēdējās kategorijas nozīme varētu pieaugt.

1.2. Funkcionalitātes prasības

Funkcionalitātes prasības (sistēmas lietojumi) veido visu korpusam nepieciešamo programmrīku kopumu:

- **navigācija** – hierarhiska korpusa tekstu atlasīšana dažādos šķērsgriezumos (pēc laika, pēc žanra, pēc autora, pēc nosaukuma utml.);
- **meklēšana** – atslēgvārda vai atslēgvārdu frāzes meklēšana korpusa saturā, kas parasti notiks arī pēc lietotāja norādītiem papildparametriem, ierobežojot aplūkojamo tekstu apgabalu (piem., pēc autora vai darba nosaukuma) vai definējot meklēšanu pēc regulārām izteiksmēm;
- **statistika** – to var iedalīt divās grupās: vārdformu biežuma statistika norādītā teksta apgabalā un statistika par sistēmu un korpusa saturu; nākotnē, attīstot tekstu augstāka līmeņa marķēšanu, nepieciešami sarežģītāki un pilnīgāki valodnieciskās statistikas rīki;
- **konkordance** – KWIC (*keyword in center*) meklēšana [McEnery] – meklētais atslēgvārds tiek izcelts vidū un tam abās pusēs tiek parādīta teksta apkaime (tiek izmantots, piemēram, vārdnīcu veidošanā un valodas apmācībā);
- **adresācija** – meklēšanas un konkordances rīki lietotājam atgriež gan atslēgvārda sastaptos kontekstus, gan norādes (URL) uz tā atrašanās vietām korpusā; adresācijas rīks pēc lietotāja pieprasījuma apstrādā šīs norādes un aizved lietotāju uz atbilstošo vietu korpusā; norādes var izveidot arī pats korpusa lietotājs;
- **tekstu prezentēšana** – jebkura pieejamā teksta vai tā fragmenta (arī, piemēram, konkordances rezultāta) automātiska sagatavošana transportējamā, prezentācijai un izdrukāšanai piemērotā formātā (PDF);
- **atgriezeniskā saite ar lietotāju** – komentāri, pētījumu rezultāti, darbu publicēšana utml.; nepieciešama redaktora kontrole; šis lietojums dotu potenciāli vērtīgu satura papildināšanas iespēju ne tikai literatūras korpusam, bet arī citu valodas funkcionālo stilu korpusiem [McEnery, Spektors];

- **tekstu marķēšana** – automātiskas tekstu satura atpazīšanas, strukturēšanas un 1. līmeņa marķēšanas rīks esošo tekstu pārnešanai uz korpusu jaunā kvalitātē un turpmākai satura papildināšanai; arī augstāka līmeņa manuālas marķēšanas rīki;
- **pieejas kontrole** – lietotāju kategoriju kontroles rīki korpusa tekstu pieejai un to izmantošanai, piemēram, interesentiem ir pieeja visiem autortiesību neaizsargātiem tekstiem, tomēr ar ierobežotu rīku funkcionalitāti atkarībā no tā, cik daudz servera resursus patērē katra operācija; savukārt akadēmiskie un komerciālie lietotāji var tikt klāt gan šiem pašiem tekstiem, gan ar individuālām pieejas tiesībām arī citiem tekstiem ar īpašu pievienoto vērtību (piem., no 320 milj. *Bank of English* vārdlietojumiem publiski pieejami ir 20 milj.) un viņi var izmantot pilnas funkcionalitātes tekstu apstrādes rīkus;
- **satura vadība** – korpusa satura administrēšanas rīki (papildināšanai, labošanai u.c.), iekļaujot marķēšanas rīku funkcionalitāti.

Visus korpusa programmrīkus var iedalīt divās kategorijās: pēc funkcionālā lietojuma un pēc lietotāju kategorijām. 1. tabulā ir parādīta abu šo kategoriju saistība un rīku izmantojums, un pēc tās uzskatāmi var secināt, ka akadēmiskie lietotāji būs nozīmīgākā korpusa auditorijas kategorija lietojumu ziņā. Līdz ar to korpusa izstrāde ir jāorientē tieši uz šo lietotāju kategoriju, savukārt pārējo divu kategoriju lietotāju funkcionālās prasības veido modificētu apakškopu akadēmiskajiem lietojumiem. Piezīme: administratīvie lietotāji ir ārpusstāvoša kategorija.

Lietotāju grupa / Lietojums	Akadēmiskie lietotāji	Komerciālie lietotāji	Interesenti	Administrācija
Navigācija	X	X	X	
Meklēšana	X		X	
Korpusa statistika	X	X	X	
Vārdformu statistika	X	X		
Konkordance	X	X	X	
Adresācija	X	X	X	
Prezentācijas formāts	X		X	
Atgriezeniskā saite	X		X	
Tekstu marķēšana	X			X
Pieejas kontrole	X	X		
Satura vadība				X

1. tabula

Pamatojoties uz secinājumiem un filoloģijas speciālistu aptaujām, sistēmas izstrādes pirmās kārtas izveide tiks orientēta uz akadēmiskajiem lietotājiem un interesentiem, realizējot būtiskākos lietojumus: navigāciju, meklēšanu, statistiku, konkordances, adresāciju un tekstu automatizētu marķēšanu. Pēdējais lietojums šajā izstrādes kārtā būs pieejams tikai administratīvajiem lietotājiem 1. līmeņa marķēšanai.

1.3. Lietotāju un korpusa interfeiss - darba vide

Korpusa realizācijas un lietotāju darba vides izvēle ir atkarīga gan no korpusa lietotāju un uzturētāju (administrācijas) prasībām, gan no tehnoloģiskajām iespējām.

Piemērotākais risinājums ir Interneta realizācija ar klienta–servera arhitektūru pretstatā lokālas darba vides realizācijai. Šādam risinājumam ir vairākas literatūras korpusam nozīmīgas priekšrocības:

- iespēja aptvert maksimāli plašāku auditoriju, veidojot funkcionējošu un vērtīgu atgriezenisko saiti ar korpusa lietotājiem;
- lietotājam nepieciešams tikai Interneta pieslēgums un pārlūkprogramma ar XML atbalstu, turklāt nepieciešamais programmnodrošinājums ir iegūstams bez maksas (piem., *Microsoft Internet Explorer*);
- vienkāršāka sistēmas uzturēšana un attīstīšana, jo viss korpuss ir centralizēts, līdz ar to nav jādomā par satura un rīku aktuālās versijas pieejamību visu kategoriju lietotājiem.

Interneta realizācijai ir arī dažas, galvenokārt tehniska rakstura, problēmas: ne visiem potenciālajiem lietotājiem būtu pieejams Interneta pieslēgums gan tīri tehniskā, gan finansiālā ziņā, kā arī ne visiem lietotājiem būtu tehniskajām prasībām atbilstoša datora konfigurācija – iespēja uzstādīt XML tehnoloģiju atbalstu. Tomēr arī lokāla risinājuma gadījumā tiktu izvirzītas noteiktas datora tehniskās konfigurācijas prasības.

Rezultātā, pamatojoties uz literatūras korpusa koncepciju (tam jābūt publiskam korpusam ar vienotu datu modeli) un uzskatot, ka tuvā nākotnē pašreizējās tehniskās problēmas visticamāk vairs nebūs problēmas, izstrāde tiks orientēta uz Interneta vidi.

2. tabulā ir parādītas Interneta risinājuma priekšrocības un problēmas dažādu robežkritēriju gadījumos.

Kritērijs	Priekšrocība	Problēma
Plaši aptverama auditorija	X	
Atgriezeniskā saite ar lietotājiem	X	
Korpusa popularitāte un tā plaša izmantojamība	X	
Centralizēts korpuss un plaša tekstu bāze	X	
Servera resursi un tīkla ātrdarbība		jānovērtē katras operācijas resursu patēriņš; nepieciešama lietotāju pieejas kontroles sistēma
Tehniskās prasības lietotājiem	nepieciešama tikai pārlūkprogramma ar XML atbalstu	tehniskie un finansiālie aspekti Interneta pieslēgumam un programmatūras nodrošinājumam
Izstrādes tehnoloģijas	X	
Korpusa uzturēšana un attīstīšana	X	
Īpaši vērtīgi un/vai autortiesību aizsargāti teksti	pievienotās vērtības popularizēšana	nepieciešama lietotāju pieejas kontroles sistēma

2. tabula

1.4. Korpusa izstrādes posmi

Par pamatu ņemot gan LU MII esošo situāciju korpusa uzkrāšanā, gan noteiktās prasības korpusa izstrādei, tika definēti sistēmas pirmās kārtas projektēšanas un realizācijas galvenie posmi:

1. Izstrādāt vienotus standartus korpusa satura aprakstīšanai, atbilstoši 1. līmeņa marķēšanai. Noteikt tekstu grupas (pēc literatūras žanriem) un izveidot tām atbilstošas datu struktūras.
2. Izstrādāt metodiku un programmrīkus tekstu automatizētai aprakstīšanai, strukturēšanai un marķēšanai atbilstoši iepriekš izveidotajiem standartiem.
3. Izveidot korpusa datu modeli un izstrādāt tā fizisko datubāzes realizāciju tekstu centralizētai glabāšanai. Uz šī pamata tiks veidoti visi literatūras korpusa programmrīki.
4. Izstrādāt korpusa lietotāju (akadēmiskā un interesentu kategorija) programmrīkus mērķa funkcionalitātes nodrošināšanai: navigācijai, meklēšanai, vārdformu statistikai, konkordancēm un adresācijai.
5. Izveidot Interneta interfeisus akadēmisko un interesentu kategoriju lietotāju saskarnei ar korpusa saturu un pieejamajiem lietojumiem.

II SISTĒMAS PROJEKTĒJUMS

2.1. Tehnoloģijas

Kā jau iepriekš tika minēts, lai nodrošinātu efektīvu korpusa tekstu apstrādi (piem., noteiktu fragmentu atlasīšanu un atainošanu) šie teksti ir strukturāli jāmarķē – jāanotē [Mason]. Tieši šī nepieciešamība rada būtisku ietekmi izstrādes tehnoloģiju izvēlē. Vispārējs *de facto* standarts tekstu marķēšanai ir SGML (*Standard Generalized Mark-up Language*). Pašlaik visi korpusa teksti ir marķēti no SGML atvasinātā formātā HTML, taču šāds marķējums apraksta tikai to, kā vizuāli atainot tekstus un tos sasaistīt (*hypertext*), nesniedzot nekādu informāciju par saturu. Tādēļ korpusa tekstu aprakstīšanai un strukturēšanai – marķēšanai, vispiemērotākā tehnoloģija ir cits SGML bāzēts formāts – XML (*eXtensible Mark-up Language*) [W3C-XML], kas valodas korpusu izstrādē, tiek lietots arī citur pasaulē (piem., Vācu valodas institūta izveidotajā *Textkorpora* [`www.ids-mannheim.de/kt/corpora.shtml`], kas aptver aptuveni 1.5 miljardus vārdformu). Šis formāts ir veidots kā vienkāršota SGML apakškopa – vienkārši lietojams un saprotams cilvēkam, un tajā pašā laikā ļoti elastīgs, efektīvs un viegli apstrādājams ar datorprogrammu palīdzību. Pie tam XML nav tikai marķēšanas standarts, bet arī vesels to atbalstošo tehnoloģiju un rīku kopums, kas nodrošina ļoti efektīvu un plašu funkcionalitāti:

- struktūras atbilstību loģiskiem likumiem (DTD gramatikas, XML shēmas);
- navigāciju datu struktūrā un noteiktu elementu atlasīšanu vai apstrādi (*XPath*);
- automātiskas transformācijas (XSLT) uz citām struktūrām, piemēram, HTML vai PDF (šeit jāpiemin, ka HTML netiek aizmirsts, bet izmantots tikai tā primārajam mērķim – datu atainošanai Internetā);
- vēl daudz citu XML apstrādes iespēju, kurām jau ir izstrādāti vai top Interneta Konsorcija standarti (piem., *XPointer*)

Salīdzinot ar tekstu strukturēšanas iespējām datubāzes tabulveida objektos, parādās vēl viena būtiska XML priekšrocība – tas ir sevišķi piemērots kokveidīgu datu struktūru aprakstīšanai un literatūras korpusa tekstu uzbūve jau pilnīgi dabīgi veido šādu struktūru. Taču tas nenozīmē, ka nav nepieciešamība pēc korpusa glabāšanas datubāzē.

Lai nodrošinātu centralizētu tekstu glabāšanu un efektīvas to atlasīšanas un meklēšanas iespējas, visi XML marķētie teksti ir jāglabā SQL datubāzes struktūrās.

Tomēr ar XML tehnoloģiju iespējām vien vēl nevar nodrošināt pilnu korpusa funkcionalitāti tekstu apstrādē. Ir nepieciešams korpusa sistēmas interfeiss Interneta vidē un programmrīki, kas, balstoties uz strukturētajiem tekstiem un izmantojot XML tehnoloģijas, lietotāju nodrošinātu ar iepriekš definētajiem lietojumiem: datu atlasīšanu un meklēšanu datu bāzē atkarībā no lietotāja norādītajiem parametriem, vārdformu statistiku konkrētā tekstu apgabalā u.c. Šim nolūkam ļoti piemērota ir programmēšanas valoda *Java* [Mason, Sun]. Tā sniedz plašu tehnoloģisko atbalstu:

- standarta klašu bibliotēkas un datu struktūras tekstu apstrādei un darbības ar regulārām izteiksmēm;
- standarta API XML datu un struktūras apstrādei (SAX un DOM);
- tehnoloģijas Interneta aplikāciju izstrādei (JSP un servleti);
- datubāzu interfeisu vaicājumu nodrošināšanā (JDBC).

Secinājums: racionāla, efektīva, maksimāli funkcionāla un paplašināma korpusa izstrādei savstarpējā sadarbībā ir jāizmanto visu augstāk aprakstīto tehnoloģiju risinājumi: tekstu aprakstīšanai, strukturēšanai, apstrādei un transformēšanai – XML; korpusa satura glabāšanai – SQL datubāze; tekstu atainošanai un lietotāju interfeisam ar korpusu – HTML un JSP; korpusa programmrīku izstrādē un interfeisam ar datubāzi – *Java* (servleti).

Galalietotāju – akadēmisko lietotāju un interesentu kategorija, datoru konfigurācijai nav nepieciešams ne *Java*, ne SQL datubāzes atbalsts – visi programmrīki un datubāze tiks darbināti uz literatūras korpusa servera. Lietotāja saņems vienīgi datus XML formātā un atbilstošas XSL definīcijas nepieciešamo datu transformēšanai HTML formātā, tāpēc vienīgā prasība ir XML un XSL atbalsts Interneta pārlūkprogrammai. To nodrošina pasaulē populārā pārlūkprogramma *Internet Explorer 5* vai jaunākas versijas. Arī *Mozilla 1.0* pārlūkprogrammā, kas kļūst arvien populārāka, tiek attīstīts šo tehnoloģiju atbalsts, savukārt *Netscape 6* un *Opera 6* atbalsta XML, bet nenodrošina XSL transformāciju iespējas. Tā kā šis ir galvenokārt nekomerciāls projekts, tad visa sistēmas izstrāde tiks orientēta uz lietotājiem, kuru programmatūra atbilst minētajiem nosacījumiem, bet pārējiem tiks piedāvāta pavisam

vienkāršota literatūras korpusa HTML versija ar ierobežotu programmrīku funkcionalitāti.

2.2. Tekstu strukturēšanas standarti – DTD gramatikas

DTD (*Document Type Definition*) ir formāla gramatika, ar kuras palīdzību definē, kādai jābūt teksta struktūrai un kādiem jābūt primitīvo elementu datu tipiem, līdz ar to, tāpat kā XML datiem, arī DTD gramatikai ir kokveida struktūra. Komplicētāka strukturēšanas gramatiku aprakstīšanas tehnoloģija ir *XML Schema*, kas dod iespēju veidot sarežģītākas datu struktūras un definēt elementiem sarežģītākus datu tipus, bet literatūras korpusa pirmās kārtas izstrādē *XML Schema* nav nepieciešama. Tomēr vēlāk, kad korpusa attīstība būs nonākusi jau līdz tekstu morfoloģiskai marķēšanai, būs nepieciešamība pāriet uz *XML Schema*.

Datiem XML marķējumā ir jāatbilst vismaz vienai no divām kvalitātes pakāpēm:

1. strukturāli pareizi marķēti dati (*well-formed*), kas atbilst minimālajiem XML sintakses nosacījumiem – katrai atverošajai iezīmei ir atbilstoša aizverošā iezīme un iezīmes ir savstarpēji pareizi ievietotas (*nested*);
2. gramatiski pareizi (*valid*) marķēti dati – pirmās pakāpes nosacījumi + iezīmes atbilst semantikai (loģiskajai struktūrai), kas norādīta DTD gramatikā.

Korpusa satura strukturēšanas un marķēšanas nepieciešamība jau tika aprakstīta augstāk – kvalitatīvs marķējums (pareizs attiecībā pret loģisko struktūru) ievērojami palielina tekstu vērtību, jo atvieglo un paplašina to funkcionālās apstrādes iespējas.

Pirms tekstu marķēšanas procesa sākšanas ir jānodefinē vienoti strukturēšanas standarti visa korpusa ietvaros. Vispirms teksti tika sadalīti pēc to veida (žanra). Esošajā LU MII literatūras korpusā ir sastopami trīs veidu žanri: dzeja, drāma un proza. Katram no šiem žanriem, sadarbojoties ar literātiem, tika izveidota atbilstoša DTD gramatika. Mērķis bija izveidot vispārējas un no literātu puses atzītas tekstu žanru struktūras.

Tā kā no iepriekšminētajiem žanriem dzejai ir vissarežģītākā struktūra, tad šīs gramatikas izveidošana bija pirmais uzdevums, lai atklātu visas problēmas, kas ir saistītas ar DTD izstrādi un to praktiskajām realizācijām, marķējot datus. Kā izrādījās, tad problēmu, ar kurām jāsaskaras, izstrādājot vienotu tekstu strukturēšanas standartu, ir

vairāk, nekā bija paredzēts, un, lai nonāktu līdz galējai dzejas DTD versijai (skatīt 1. shēmu) bija jāveic četras iterācijas.

```
<!-- Dzejas krājuma saknes elements -->
<!ELEMENT dzeja (virsraksts,apaksvirsraksts?,iespraudums*,nodala+)>
<!ATTLIST dzeja
    autors CDATA #REQUIRED
>

<!-- Krājuma nodaļa -->
<!ELEMENT nodala (
    iespraudums*,virsraksts?,apaksvirsraksts*,
    (iespraudums*,(dzejolis|proza|nodala)+,iespraudums*)+
)>

<!-- Dzejolis krājuma nodaļā -->
<!ELEMENT dzejolis (
    virsraksts?,apaksvirsraksts*,
    (iespraudums*,(pants|dzejolis)+,iespraudums*)+
)>

<!-- Krājuma, nodaļas vai dzejoļa virsraksts -->
<!ELEMENT virsraksts (#PCDATA)>

<!-- Virsraksta paplašinājums (paskaidrojums) -->
<!ELEMENT apaksvirsraksts (#PCDATA)>

<!-- Dzejoļa pants -->
<!ELEMENT pants (rinda)+>

<!-- Fragments (moto, citāts, vēlējums u.c.), kas nav dzejolis -->
<!ELEMENT iespraudums (rinda)+>

<!-- Panta vai iesprauduma rinda -->
<!ELEMENT rinda (#PCDATA)>

<!-- Prozas fragments, kas nav dzeja prozā -->
<!ELEMENT proza (rindkopa)+>

<!-- Rindkopa prozas fragmentā -->
<!ELEMENT rindkopa (#PCDATA)>
```

1. shēma

Sākotnēji izstrādātā dzejas struktūra bija ļoti detalizēta, taču reālajā situācijā, izstrādājot automatizētu marķēšanas rīku, izrādījās, ka ir ļoti problemātiski iegūt DTD atbilstošu tekstu XML marķējumā (skatīt nākošo nodaļu). Galvenokārt šī problēma ir saistīta ar esošā korpusa neviennozīmīgo HTML marķējumu (Interneta lapu veidošanai ir izmantots *Microsoft Front Page*) un dzejas struktūras sarežģītību. Pie tam ir jārēķinās, ka ar šādām problēmām nāksies saskarties arī papildinot korpusu ar jauniem tekstiem, ne tikai apstrādājot jau esošos tekstus. Tādēļ labāks risinājums pirmajā strukturēšanas kārtā ir vienkāršot DTD, atkāpjoties no maksimālas detalizācijas pakāpes. Savukārt

nākošās detalizācijas iterācijas (gan automātiskas, gan manuālas), jau būs iespējams veikt vienkāršāk un racionālāk, par pamatu ņemot iepriekšējā kārtā strukturētos tekstus. Un apspriežot šo problēmu ar literatūras un korpusa speciālistiem, izrādījās, ka sākumā nav nemaz nepieciešama tik sarežģīta struktūra. Piemēram, elements `iespraudums` ir trīs līdzīgu elementu – moto, citāta un vēlējuma – vienkāršojums, jo radās lielas problēmas tos automātiski atšķirt HTML marķētajā tekstā. Bez tam, tā kā šāda veida elementi no kopējā dzejas teksta sastāda pavisam nelielu daļu (piem., populārākajos Raiņa krājumos ir vidēji 8 iespraudumi), tad nav lietderīgi neracionāli sarežģīt automātiskās marķēšana programmu un veltīt tam vairāk laika nekā būtu nepieciešams manuāli marķējot šos elementus.

Secinājums: ja rodas problēmas veicot automātisku tekstu strukturēšanu, ieteicams šo procesu sadalīt vairākās iterācijās, katrā no tām nonākot pie detalizētākas struktūras. Literatūras korpusa gadījumā detalizācijas pakāpei pirmajā izstrādes posmā bija jāatrod kompromiss, kas apmierinātu funkcionalitātes prasības no literātu puses un būtu iespējams realizēt veicot automātisku marķēšanu.

Šeit arī jāpiemin vēl viena XML tehnoloģiju priekšrocība. Ja rodas nepieciešamība mainīt tekstu struktūru vai arī tikai elementu nosaukumus, kas nozīmē izmaiņu veikšanu arī pašos korpusa marķēšanas pamatos – DTD gramatikā, tad šo izmaiņu veikšana jau nomarķētajam korpusa saturam ir vieglāk realizējama ar XSL transformāciju palīdzību, salīdzinot, piemēram, ar automatizēta tekstu strukturēšanas programmrīka izstrādāšanu (skatīt nākošo sadaļu) šim mērķim, vai tabulu struktūras maiņu pilnīgas datubāzes realizācijas gadījumā.

Balstoties uz iegūto pieredzi, nākošā gramatika tika izveidota prozai, kas struktūras ziņā ir vienkāršāka par dzeju (skatīt 2. shēmu).

Izstrādājot saturu aprakstošas struktūras modeli, var rasties problēmas nosakot robežu starp tīri saturisku marķēšanu un vizuālās struktūras elementu izmantošanu. Šajā gadījumā, definējot iesprauduma elementu, tā struktūrā nācās iekļaut rindas pārnesumu, jo iespraudums prozā var būt ne tikai teksta rindkopa (arī prozā), bet arī, piemēram, dzejolis. Tā kā literatūras korpusa lietojamība ir paredzēta ne tikai valodnieciskiem pētījumiem, kuros tekstu vizuālā struktūra nav svarīga, tad saturiskā marķējuma atainošanu nepieciešams nodrošināt maksimāli tuvinātu oriģinālajai darba prezentācijas struktūrai.

```
<!-- Prozas darba saknes elements -->
<!ELEMENT proza (virsraksts,apaksvirsraksts?,iespraudums*,nodala+)>
<!ATTLIST proza autors CDATA #REQUIRED>

<!-- Darba nodaļa -->
<!ELEMENT nodala (
    virsraksts?,apaksvirsraksts*,
    (iespraudums*,(rindkopa|nodala)+,iespraudums*)+
)>

<!-- Darba vai nodaļas virsraksts -->
<!ELEMENT virsraksts (#PCDATA)>

<!-- Virsraksta paplašinājums (paskaidrojums) -->
<!ELEMENT apaksvirsraksts (#PCDATA)>

<!-- Teksta rindkopa nodaļā -->
<!ELEMENT rindkopa (#PCDATA)>

<!-- Dzejas vai teksta fragments, kas tieši neattiecas uz darbu -->
<!ELEMENT iespraudums (#PCDATA|parnesums)*>

<!-- Rindas pārnesums (vizuālā noformējuma elements) -->
<!ELEMENT parnesums EMPTY>
```

2. shēma

Savukārt, veidojot DTD gramatiku drāmām, par pamatu tika ņemts Džona Bosaka (Jon Bosak) *Shakespeare DTD*. Šo Šekspīra lugu strukturēšanas standartu ir izveidojis Interneta Konsorcija XML standartu izstrādes un attīstības grupas vadītājs, un tas tiek daudzviet izmantots kā lugu marķējuma paraugs. Sadarbībā ar literatūras speciālistiem, izpētot Bosaka izveidotās struktūras pielāgojamību latviešu dramaturgu lugām, atklājās tuva struktūru atbilstība – nepieciešams bija modificēt vai izmest dažus elementus. Atvasinātā drāmas darbu marķēšanas gramatika ir definēta 3. shēmā.

Veidojot literatūras korpusa tekstu strukturēšanas standartus, darba autors pētīja arī pasaulē izstrādātos principus un to realizācijas piemērus. Vieni no atzītākajiem standartiem balansētu valodas korpusu marķēšanas pamatā ir:

- TEI (*Text Encoding Initiative*) – tekstu vispārīga aprakstīšana (ietverot metadatus), lai nodrošinātu apmaiņu un apstrādi starp dažādām sistēmām;
- CES (*Corpus Encoding Standard*) – minimālās tekstu strukturēšanas prasības, lai korpuss atbilstu 1. marķēšanas līmenim.

Šie standarti apraksta vienotas, vispārīgas tekstu pamatstruktūras un marķēšanas vadlīnijas, taču iekšējie elementi var tikt detalizēti pēc katras sistēmas vajadzībām.

Izstrādes pirmajā kārtā korpusa tekstiem ir izveidotas tikai specifiskās struktūras (atbilstošas 1. marķēšanas līmenim), taču, rodoties nepieciešamībai iekļauties vispārīgos standartos, tās var automatizēti apstrādāt, "ietinot" standartizētās pamatstruktūrās.

```
<!-- Drāmas saknes elements -->
<!ELEMENT drama (
    virsraksts,apaksvirsraksts?,personazs,remarka,
    ievads?,prologs?,celiens+,epilogs?
)>
<!ATTLIST drama autors CDATA #REQUIRED>

<!-- Drāmas, ievada, cēliena, skata, epiloga vai prologa
     virsraksts -->
<!ELEMENT virsraksts (#PCDATA)>

<!-- Drāmas, ievada, cēliena, epiloga, prologa vai skata
     apakšvirsraksts -->
<!ELEMENT apaksvirsraksts (#PCDATA)>

<!-- Darbojošos personu saraksts (drāmas sākumā) -->
<!ELEMENT personazs (persona|grupa)+)>

<!-- Personu grupa ar kādu kopīgu pazīmi -->
<!ELEMENT grupa (persona+,remarka)>

<!-Atsevišķa persona (arī vispārīgu pers. kopums, piem., statisti) -->
<!ELEMENT persona (#PCDATA)>

<!-- Drāmas remarka (vispārējs izvietojuma un darbības apraksts) -->
<!ELEMENT remarka (#PCDATA)>

<!-- Drāmas ievads (aiz pers. saraksta, pirms 1. cēliena) -->
<!ELEMENT ievads (
    virsraksts,apaksvirsraksts*,(skats+|(runa|remarka)+)
)>

<!-- Drāmas cēliens -->
<!ELEMENT celiens (
    virsraksts,apaksvirsraksts*,prologs?,skats+,epilogs?
)>

<!-- Ievada vai cēliena skats -->
<!ELEMENT skats (virsraksts,apaksvirsraksts*,(runa|remarka)+)>

<!-- Drāmas vai cēliena prologs -->
<!ELEMENT prologs (virsraksts,apaksvirsraksts*,(runa|remarka)+)>

<!-- Drāmas vai cēliena epilogs -->
<!ELEMENT epilogs (virsraksts,apaksvirsraksts*,(runa|remarka)+)>

<!-- Runa, ko vienā reizē saka persona -->
<!ELEMENT runa (persona+,(rinda|remarka)+)>

<!-- Viena runas rinda -->
<!ELEMENT rinda (#PCDATA)>

<!-- Skatuves un/vai darbības apraksts -->
<!ELEMENT remarka (#PCDATA)>
```

3. shēma

2.3. Tekstu metainformācija

Iepriekšējā nodaļā ir definētas autora izveidotās gramatikas, pēc kurām vadoties notiek visu literatūras korpusa tekstu strukturēšana un aprakstīšana. Šīs struktūras attiecas tikai uz tekstu saturu un neiekļauj nekādu netiešu, bet saistītu un aprakstošu informāciju par šo saturu, piemēram, darbu autorus, žanrus, publicēšanas datumus, autortiesības utt. Tomēr, lai nodrošinātu pilnvērtīgas tekstu navigācijas, meklēšanas un atlases iespējas, nepieciešams paralēli uzkrāt arī ar šiem tekstiem saistīto informāciju – metainformāciju jeb datus par tekstiem. Tādēļ nākošais izstrādes posms pēc DTD gramatiku definēšanas ir vienota standarta metadatu struktūras izveidošana. Šeit gan jāpiebilst, ka arī DTD definētās struktūras ir metadati, tikai tie ir ļoti detalizēti un attiecas tieši uz katru konkrēto teksta fragmentu, nevis uz visu darbu kopumā.

Pirmais solis bija aprakstošās informācijas noteikšana visu žanru tekstiem, apzinot visu iespējamo saistīto informāciju, kas pēc dažādu veidu kritērijiem identificē tekstus (autora un publicētāja dati, teksta žanrs, īss satura apraksts un atslēgvārdi u.c.), lai nodrošinātu pēc iespējas rezultatīvāku meklēšanu. Pēc tam šī metainformācijas kopa tika izvērtēta un saklasificēta, atlasot reāli nepieciešamos informācijas elementus.

Otrais solis (pašreizējā stadija) ir metadatu aprakstīšanas standarta izveide un praktiskā realizācija. Šajā gadījumā bija jāizšķiras par konceptuālu jautājumu: vai nu literatūras korpusam veidot savu metadatu struktūras standartu, vai arī par pamatu ņemt vispārējus standartus. Pirmajā gadījumā detalizēta meklēšana būtu iespējama tikai strādājot ar literatūras korpusa rīkiem, bet otrajā gadījumā būtu nodrošināta savietojamība un metadatu struktūras saprotamība arī dažādiem Interneta meklēšanas servisiem, kas atbalstītu izvēlēto vispārējo standartu. Attīstoties Interneta robotiem, šāda izvēle varētu arī novest pie tā, ka tekstus indeksētu un meklētu no citiem portāliem, nevis tikai izmantojot korpusa programmrīkus, tomēr liels ieguvums būtu ievērojama auditorijas paplašināšana, jo potenciālie lietotāji pat nezinot par literatūras korpusa eksistenci, meklējot pēc metadatu kritērijiem, varētu to atrast, izmantojot kādu no Interneta meklēšanas servisiem (piem., `www.google.com`).

Literatūras korpusa tekstu metadatu aprakstīšanas sistēma tiek veidota pamatojoties uz DC (*Dublin Core*) standartu [DCMI], lai nodrošinātu savietojamību ar jebkuru Interneta meklēšanas rīku, kurš "saprot" šo metadatu struktūru. DC ir plaši atzīts standarts un DCMI sadarbojas ar daudzām Interneta standartu izstrādes

organizācijām, piemēram, W3C. DC pamatā ir 15 elementu kopa (*DC Metadata Element Set*) no kuras tika izvēlēti literatūras korpusam nepieciešamie metadatu elementi:

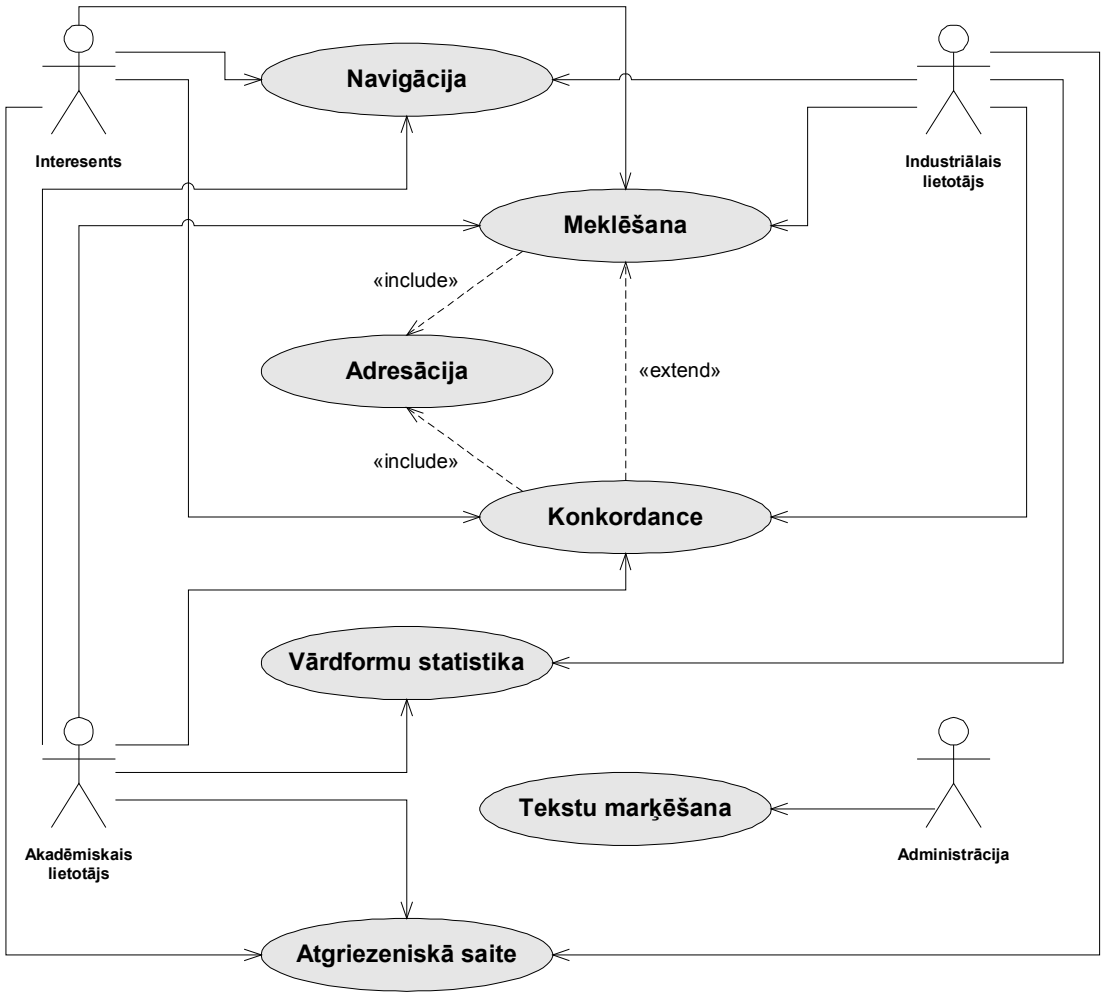
- *title* – resursa nosaukums (literārā darba virsraksts);
- *creator* – resursa satura pirmautors (teksta autors);
- *subject* – saturā aptvertā tematika, atslēgvārdi;
- *description* – konspektīvs satura apraksts;
- *publisher* – entīte (persona, organizācija vai serviss), kas nodrošinājusi satura pieejamību un pievienoto vērtību (korpusa izstrādes organizācija);
- *date* – datums, saistīts ar resursa "dzīves ciklu" (teksta pievienošana korpusā);
- *type* – satura (teksta) žanrs;
- *identifier* – resursu viennozīmīgi identificējoša norāde (URI);
- *source* – atsauce uz resursu no kura ir atvasināts saturs (izdevniecības dati);
- *language* – valoda, kādā saturs ir reprezentēts (latviešu autoriem ir sastopami atsevišķi darbi arī svešvalodās);
- *relation* – atsauces uz saistītiem resursiem (DTD gramatika, autora biogrāfija);
- *rights* – informācija par satura un pievienotās vērtības autortiesībām.

Veidojot metadatu struktūru, parādījās interesants jautājums: darba autora vārds ir teksta meta informācija, vai neatdalāms satura elements? Datu dublēšana, protams, nav ieteicams risinājums, tomēr šajā situācijā pareizāk būtu abās vietās iekļaut norādi uz autoru: metadatos, lai nodrošinātu pilnvērtīgu meklēšanu, bet teksta struktūrā – lai nezaudētu nozīmīgu saturam piederošu elementu gadījumā, ja metadati nav pieejami.

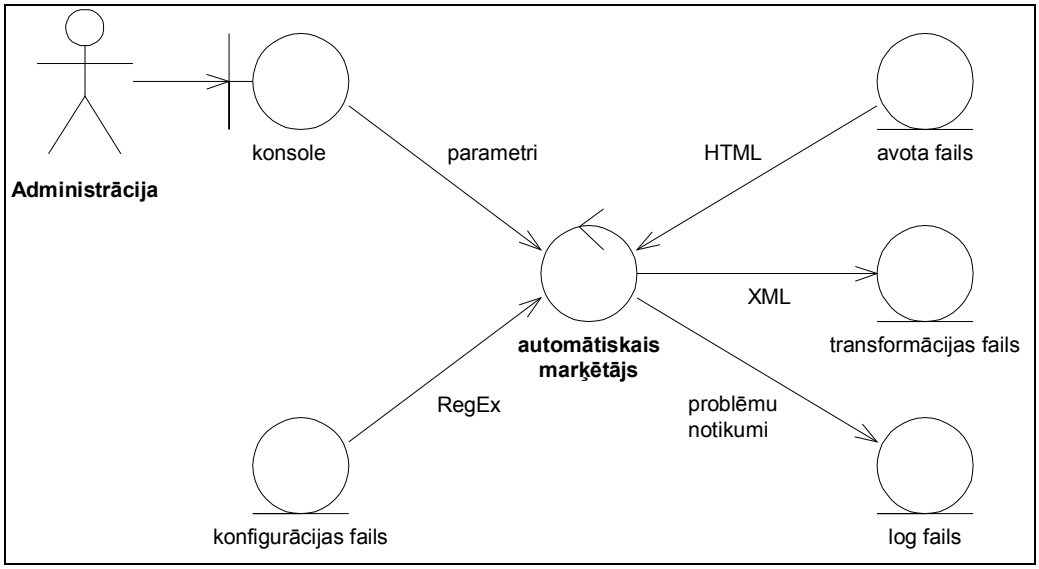
2.4. Lietojumu diagrammas

Tā kā literatūras korpusa projektējums vēl ir veidošanas procesā, šajā sadaļā ir vispārīgi ir parādīti izstrādes pirmajā kārtā realizējamie lietojumi (*use-cases*) kopējās sistēmas ietvaros (skatīt 1. diagrammu). Kā jau iepriekš tika minēts, šajā kārtā tiks realizēti būtiskākie korpusa programmrīki, bet vēlāk sistēma tiks attīstīta tālāk.

Detalizēti ir atainots šobrīd vienīgais realizētais lietojums – tekstu automatizētās strukturēšanas un marķēšanas rīks (skatīt 2. diagrammu). Sīkāk šī lietojuma projektējums un realizācija ir aprakstīti nākošajā nodaļā.



1. diagramma



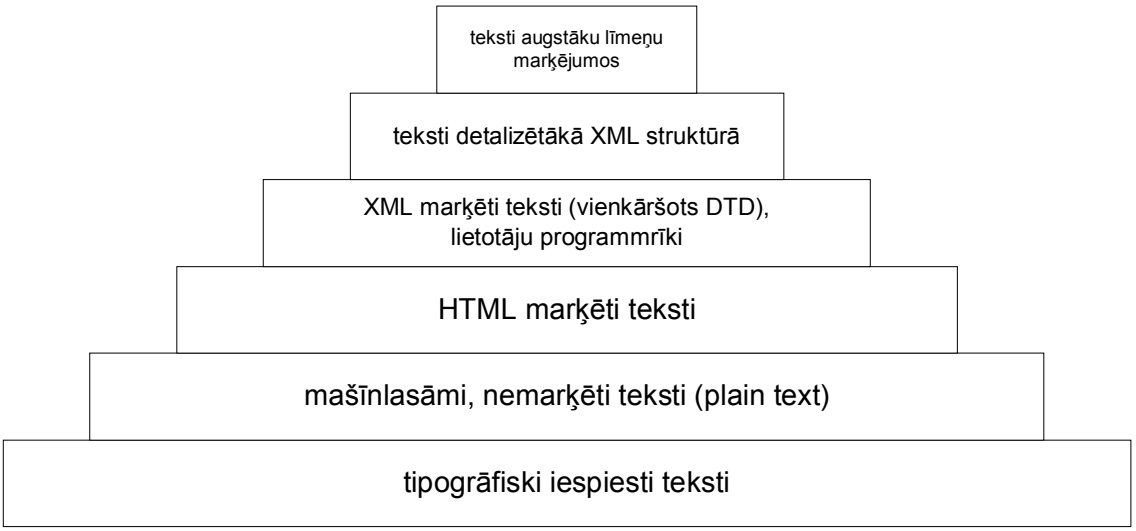
2. diagramma

III KORPUSA PROGRAMMRĪKU IZSTRĀDE

3.1. Automātiska tekstu strukturēšana XML formātā

Tā kā visu mērķauditorijas lietotāju programmrīku izstrāde ir atkarīga no literatūras korpusa satura struktūras, tad vispirms nepieciešams transformēt esošo tekstu masīvu jaunā projektējuma datu modelī (skatīt iepriekš sadaļu *DTD gramatikas*). Tradicionāli informatīvo sistēmu izstrādē datu pārnešana no vecās sistēmas uz jauno notiek izstrādes beigu fāzē – jaunās sistēmas ieviešanā ("ūdenskrituma" modelis), tomēr šajā gadījumā tekstu transformēšanu un marķēšanu bija jāveic paralēli korpusa strukturēšanas standartu izveidei. Šāda risinājuma izvēle ir saistīta ar to, ka veidojot programmrīku automātiskām transformācijām, bija paredzama problēmu atklāšana sākotnēji izveidotās struktūras sasniegšanā, līdz ar to rastos nepieciešamība meklēt kompromisus un mainīt pirmajā iterācijā izstrādāto datu modeli, lai realizētu maksimālu automatizācijas pakāpi, tajā pašā laikā neatkāpjoties no būtiskiem struktūras elementiem. Tādejādi DTD gramatiku un automātiskās marķēšanas rīka attīstība notika pēc "spirāles" veida modeļa – sākotnējā struktūra kalpoja par pamatu marķētāja pirmās versijas izstrādē, savukārt, balstoties uz testēšanas laikā atklātajām automatizācijas problēmām un DTD nepilnībām, tika mainītas atbilstošās struktūras (papildinot ar trūkstošiem elementiem, vienkāršojot programmiski neatpazīstamus elementus utml. – skatīt zemāk).

Lai ātrāk apzinātu DTD projektējumu nepilnības un iespējamās problēmas nepieciešams veikt masveidīgu marķēšanu, aptverot ar dažādas sarežģītības tekstu kopas (piem., dzejas marķēšanas procedūra tika veidota balstoties uz desmit pazīstamākajiem Raiņa krājumiem). Tas būtu arī pamats uz kura tālāk veikt programmrīku izstrādi galalietotājiem, turklāt sākumā nav nepieciešami saturiski pilnīgi pareizi transformēti teksti – tiem ir jābūt vismaz gramatiski pareiziem (*valid*) attiecībā pret DTD. Arī sarežģītākas un detalizētākas tekstu struktūras un to saturisko pareizību, kā jau iepriekš darbā tika minēts, ir vieglāk nodrošināt sākumā iegūstot vienkāršāku struktūru un tad to pakāpeniski attīstot (piem., dzejas elementi moto, citāts un vēlējums sākotnēji tiks uzskatīti par vienu elementu iespraudums). Literatūras korpusa esošās un nākotnē paredzētās tekstu marķējumu pakāpes ir uzskatāmāki atainotas 3. diagrammā (katra augstākā pakāpe ietver zemākās pakāpes funkcionalitāti).

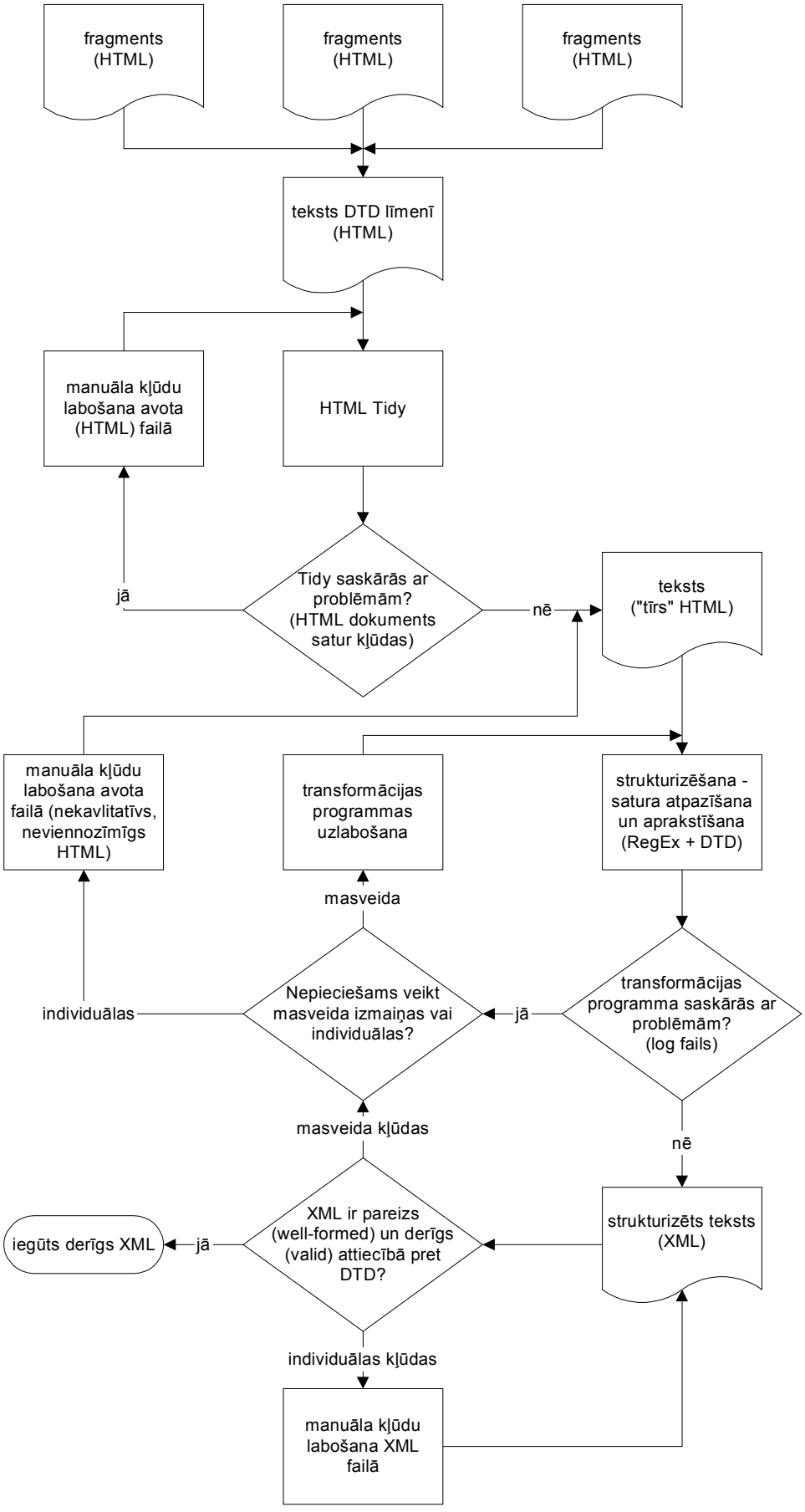


3. diagramma

3.1.1. Metodika

Visu uzkrāto tekstu 1. līmeņa strukturēšanas un marķēšanas procesu pilnīgi automatizēt praktiski nav iespējams, kā arī ne visu procesa soļu automatizācijai ir lietderīgi tērēt laiku (piem., tekstu fragmentu apvienošana viena literāra darba ietvaros – DTD līmenī). Lai transformācijas procesu padarītu standartizētu un līdz ar to efektīgāku, radās vajadzība izveidot marķēšanas metodiku. Galvenais nosacījums bija, lai administratīvie lietotāji, galvenokārt ar filoloģisku specialitāti, varētu vienkārši veikt tekstu transformāciju. Sākotnēji tika izstrādāts šī lietojuma programmas kodols, kas vēlāk tika pilnveidots attīstot interfeisu un padarot šī rīka lietojamību universālu literatūras korpusa ietvaros. Tā rezultātā arī transformācijas process kļuva vienkāršāks (skatīt 4. diagrammu) un to var iedalīt 3 posmos, kurus iteratīvi var būt vajadzība atkārtot:

- tekstu fragmentu atlasīšana un apvienošana viena literārā darba ietvaros, nepieciešamo tukšo iezīmju salikšana (manuāli) un esošā HTML marķējuma "tīrīšana" (automātiski); kļūdaina HTML marķējuma gadījumā tā labošana;
- regulāru izteiksmju sastādīšana satura elementu programmiskai atpazīšanai (manuāli) un teksta strukturēšana un marķēšana izmantojot izstrādāto programmrīku (automātiski);
- iegūtā rezultāta – XML marķēta teksta – pārbaude attiecībā pret DTD gramatiku (automātiski); nepieciešamības gadījumā marķējuma rediģēšana vai regulāro izteiksmju pilnveidošana (manuāli) un atkārtota teksta strukturēšana.


fragments (HTML)
fragments (HTML)
fragments (HTML)
teksts DTD līmenī (HTML)
manuāla kļūdu labošana avota (HTML) failā
HTML Tidy
jā
Tidy saskārās ar problēmām? (HTML dokuments satur kļūdas)
nē
teksts ("tīrs" HTML)
manuāla kļūdu labošana avota failā (nekavlitatīvs, neviennozīmīgs HTML)
transformācijas programmas uzlabošana
strukturizēšana - satura atpazīšana un aprakstīšana (RegEx + DTD)
masveida
individuālas
Nepieciešams veikt masveida izmaiņas vai individuālas?
jā
transformācijas programma saskārās ar problēmām? (log fails)
masveida kļūdas
nē
iegūts derīgs XML
jā
XML ir pareizs (well-formed) un derīgs (valid) attiecībā pret DTD?
strukturizēts teksts (XML)
individuālas kļūdas
manuāla kļūdu labošana XML failā

4. diagramma

3.1.2. HTML tīrīšana

Marķēšanas procesa pirmajā posmā pēc tekstu fragmentu apvienošanas attiecīgā darba lielākajā loģiskajā struktūrā, nepieciešams "tīrīt" tā HTML marķējumu. Tas nozīmē atrast visas strukturālās kļūdas (piem., izlaistas elementu aizverošās iezīmes vai nepareizi iekļauti elementi), izmest strukturāli nevajadzīgas iezīmes, kas, piemēram, var rasties, ja marķētais teksts ir iegūts ar *Microsoft Word* vai *Front Page* Interneta lapu izstrādes līdzekļiem (šeit gan jāuzmanās, lai netiktu izmestas satura atpazīšanai nozīmīgas iezīmes), kā arī viennozīmīgi noformēt pašu marķējumu (normalizēt atstarpes starp iezīmēm un iezīmju nosaukumu reģistrjūtību utml.). Tīrs HTML marķējums atvieglo regulāro izteiksmju sastādīšanu, kas nepieciešamas teksta struktūru automatizētai atpazīšanai, kā arī nodrošina kvalitatīvāku transformācijas rezultātu, jo tiek uzlabots HTML marķētā teksta viennozīmīgums.

Esošā literatūras korpusa transformācijā šīs procedūras automātiskai veikšanai ļoti piemērots ir jau gatavs risinājums – W3C izstrādātais programmrīks *HTML Tidy* [W3C-Tidy]. Strādājot ar šo rīku, nepieciešams norādīt tikai HTML failu un tīrīšanas nosacījumus, tomēr, lietojot dažādas šo nosacījumu kombinācijas, iespējams iegūt būtiski atšķirīgus rezultātus. *Tidy* specifikācijā nosakot iespējami nepieciešamos parametrus un veicot vairākus testus papildus noklusētajiem parametriem tika atrasta piemērotākā nosacījumu kombinācija (tikai divi papildus nosacījumi):

- `wrap int`: *int* ir skaitlis, kas lielāks par iespējami garāko rindiņu (pēc simbolu skaita) HTML failā – mākslīgu rindas pārnesumu veidošanas novēršana;
- `raw`: simbolus, kuru kods ir lielāks par 127 (ne ASCII simboli) netranslēt par atbilstošajām entītēm – latviešu valodas burtu saglabāšana tekstā.

Testēšanas laikā atklājās arī vairākas problēmas, kuras tika ņemtas vērā marķēšanas metodikas un programmrīka izstrādes veidošanā:

- *Tidy* izmet tukšos elementus, līdz ar to, piemēram, tukšais dzejoļa virsraksts `<p align="center"></p>` vietā ir jāiezīmē kā `<p align="center"> </p>` (izmests tiks tikai ` `);
- Ne vienmēr pēc strukturālo kļūdu izlabošanas saglabājas marķējuma noformējuma viennozīmīgums (kļūdainās vietas ir jāpārbauda);
- *Tidy* oriģinālā (2000. gada) versija strādā korektāk par uzlabojumiem.

3.1.3. Automātiska marķēšana

Pēc "attīrīta" marķētā teksta iegūšanas nākošais transformācijas posms ir šī teksta strukturēšana, izmantojot izveidoto automātiskās marķēšanas rīku. Tā kā esošā teksta vizuālajā struktūrā ir sagaidāms samērā konsekvents marķējuma noformējums, tad tekstu satura atpazīšana tiek nodrošināta ar *Java* atbalstītu regulāro izteiksmju tehnoloģijas palīdzību. Korpusa administratoram, kas veic transformāciju, ir jāmeklē kopsakarības šajā strukturālajā noformējumā, pēc kurām var atpazīt atbilstošā žanra elementu robežas. Atklātās kopsakarības ir jāapraksta ar regulārām izteiksmēm un jānodod kā parametri automātiskās marķēšanas rīkam. Interfeiss tiek nodrošināts ar *Windows* aplikācijas palīdzību, kas norādi uz transformējamā teksta failu un lietotāja nodefinētās regulārās izteiksmes saglabā marķētāja konfigurācijas failā un izsauc atbilstošā žanra marķēšanas klasi. Šis specializētais marķētājs tad secīgi pa rindiņai lasa visu HTML marķēto tekstu un pēc nodefinētajām regulārajām izteiksmēm mēģina noteikt elementu robežas (tas tiek nodrošināts ar *Java 1.4* standartbibliotēkas klašu `java.regex.Pattern` un `java.regex.Matcher` palīdzību). Atrodot kādu elementu, tam tiek "nogrieztas" HTML marķējuma iezīmes, tādejādi iegūstot pilnīgi nemarķētu teksta fragmentu, kas tiek "ietīts" atbilstošā XML marķējumā un ievietots jaunajā teksta struktūrā. Rezultātā lietotājam tiek atgriezts saturiski strukturēts teksts un *log* fails ar visiem marķēšanas problēmu notikumiem.

Tomēr marķētājs nav tikai teksta virknīšu meklēšanas un aizvietošanas rīks – ar iebūvētu likumu palīdzību, kas balstīti uz apgrieztas atbilstošās DTD gramatikas, tiek kontrolēta struktūras loģiskā pareizība (datora atmiņā tiek glabāts steks ar tekošajā situācijā atvērtajiem struktūras elementiem). Ar apgrieztu DTD gramatiku ir domāti likumi, kas nosaka kādi elementi drīkst atrasties struktūras koka zarā pirms katra no elementiem (noteikti tiek tikai tie elementi, kas ir par vienu līmeni seklāki nekā aplūkojamais). 4. shēmā ir parādīts piemērs ar apgrieztu dzejas gramatiku.

```
rinda          ← (pants|iespraudums)
rindkopa       ← (proza)
proza          ← (nodala)
pants          ← (dzejolis)
iespraudums    ← (dzejolis|nodala|krajums)
virsraksts     ← (dzejolis|nodala|krajums)
apaksvirsraksts ← (dzejolis|nodala|krajums)
dzejolis       ← (dzejolis|nodala)
nodala         ← (nodala|krajums)
```

4. shēma

Cita alternatīva regulāro izteiksmju pieejai ir DOM (*Document Object Model*) ar kura palīdzību visu HTML koku var ielasīt atmiņā un dažādos šķērsgriezumos apstaigāt, taču šāds risinājums būtu sarežģītāks, jo tekstu saturu esošajā marķējumā ir vieglāk automatizēti atpazīt analizējot iezīmju kontekstu, satura rakstzīmju izmantojumu un to reģistrjūtību.

Piemēram, zemāk doto HTML marķējuma fragmentu automātiskais marķētājs apstrādās šādi: vispirms tiks nolasītas centrēta paragrāfa iezīmes starp kurām atrodas teksts, kas veidots tikai no lielajiem burtiem; salīdzinot ar regulāro izteiksmju kopu šī rindiņa tiks atpazīta kā dzejoļa virsraksts un marķētājs transformācijas struktūrā (XML) atvērs dzejoļa elementu ierakstot tajā arī virsraksta elementu; pēc tam tiks atvērts pants un ierakstīta pirmā rinda, atpazīstot panta sākumu; pirmajai rindai seko otra rindiņa un tā kā panta elements jau ir atvērts, tad tiks ierakstīts tikai panta rindas elements; ceturtā HTML rindiņa atkal tiks atpazīta kā panta sākums, bet tā kā jau viens pants ir atvērts un saskaņā ar DTD viens panta elements nevar ietvert citu panta elementu, tad atvērtais pants tiks aizvērts, atverot jauno.

```
...
<p align="center">AGRI NO RĪTA</p>
<p align="center"> Guļ manā priekšā <br>
Pilsēta, grimusi</p>
<p align="center">Riebjas man nokāpt<br>
...
```

Rezultātā marķētājs būs izveidojis šādu struktūru:

```
...
<dzejolis>
<virsraksts>AGRI NO RĪTA</virsraksts>
<pants>
<rinda>Guļ manā priekšā</rinda>
<rinda>Pilsēta, grimusi</rinda>
</pants>
<pants>
<rinda>Riebjas man nokāpt</rinda>
...
```

Loģiskās struktūras nodrošināšana, atbilstoši specifiskiem DTD, visticamāk ir problēma, kādēļ nav pieejami universāli tekstu strukturēšanas rīki.

Literatūras korpusa automātiskās marķēšanas rīka izstrādes problēmas bija galvenokārt saistītas ar neviennozīmīgo HTML marķējumu (piem., iesprauduma elements dzejā vietām bija marķēts kā lapas labajai malai piesaistīts paragrāfs, citur kā

centrēts paragrāfs, kuru sarežģīti atšķirt no panta) un grūtībām atpazīt atsevišķus struktūras elementus (piem., moto no vēlējuma, jo tie esošajā marķējumā ir vienādi iezīmēti, kā rezultātā bija jāvienkāršo dzejas DTD). Atklājās arī nepieciešamība atsevišķos gadījumos HTML marķējumam pievienot "mākslīgas" iezīmes, piemēram, tukšu nodaļas virsrakstu, ja dzejoļu krājums satur nodaļu bez virsraksta, savādāk nav iespējams automātiski noteikt šo robežu. Minēto problēmu rezultātā par primāro prasību tika izvirzīta tekstu transformāciju loģisko struktūru pareizība (*validation*) attiecībā pret DTD. Savukārt loģiskās struktūras saturiskās pareizības pārbaudi un atsevišķu elementu detalizāciju būs vienkāršāk un efektīvāk nodrošinot nākošajā, daļēji automātiskās marķēšanas iterācijā, izmantojot cilvēka intelektu lēmumu pieņemšanā.

NOBEIGUMS

Pašlaik notiek darbs pie literatūras korpusa pirmās kārtas projektējuma detalizētas un pilnīgas izstrādes, kas turpmāk tiks ņemts par pamatu korpusa programmrīku tehniskajā realizācijā. Apzinātas ir arī galvenās izstrādes problēmas, vēlamais rezultāts.

Praktiskās realizācijas līmenī šobrīd notiek darbs pie automātiskā marķētāja interfeisa uzlabošanas, lai lietotājiem ar to būtu maksimāli vienkārši strādāt, jo arī nākotnē lielākā daļa korpusam pievienojamo tekstu būs transformējami no "ārējām" marķējumu struktūrām. Tuvākajā laikā, pabeidzot sistēmas projektēšanu, tiks sākts darbs pie korpusa datubāzes un mērķauditorijas interneta rīku izstrādes.

Turpmākās izstrādes gaita, problēmas un sasniegtie galarezultāti tiks pilnībā aprakstīti bakalaura darbā. Iegūtās zināšanas un pieredze tiks izmantotas arī tālākā darbā, veidojot latviešu valodas korpusu.

SAĪSINĀJUMU UN TERMINU SKAIDROJUMI

API – *Aplication Programming Interface* – programmējams aplikācijas interfeiss.

CES – *Corpus Encoding Standard* – minimālās tekstu marķēšanas prasības (atbilstoši 1. līmenim), kas jāsasniedz korpusa izstrādē, lai to varētu uzskatīt par standartizēti strukturētu korpusu [`http://www.cs.vassar.edu/CES`].

DC – *Doublin Core* – DCMI izstrādāts metadatu strukturēšanas un aprakstīšanas standarts (IETF RFC 2413).

DCMI – *Dublin Core Metadata Initiative* – organizācija, kas nodarbojas ar vispārēju metadatu standartu izstrādāšanu plašām lietojumu sfērām Internetā [`http://dublincore.org`].

DOM – *Document Object Model* – no platformas un programmēšanas valodas neatkarīgs interfeiss, kas ļauj programmai dinamiski piekļūt dokumenta saturam, struktūrai un stilam un mainīt tos [W3C-DOM].

DTD – *Document Type Definition* – formāla gramatika XML dokumentu datu loģiskās struktūras un elementu datu tipu definēšanai.

Java – *Sun Microsystems* izstrādāta objektorientēta, platformneatkarīga, programmēšanas valoda [Sun-J2SE].

JDBC – *Java DataBase Connectivity* – API, kas ļauj piekļūt jebkuram tabulāram datu avotam no *Java* programmēšanas valodas. Nodrošina piekļuvi dažādu SQL DBVS datubāzēm.

JSP – *JavaServer Pages* – tehnoloģija, kas dod iespēju ātri izstrādāt un viegli uzturēt, dinamiskas Interneta lapas ar apjomīgu informāciju [Sun-J2EE].

HTML – *Hyper Text Mark-up Language* – vienkārša, ierobežota hipertekstu dokumentu marķēšanas valoda, SGML apakškopa. Plaši izmantota Interneta lapu veidošanā. Pamatā paredzēta dokumentu vizuālās struktūras aprakstīšanai.

OCR – *Optical Character Recognition* – optiska simbolu atpazīšana. Tekstu skenēšanas un automātiskas atpazīšanas tehnoloģija.

PDF – *Portable Document Format – Adobe Systems* izstrādāts universāls platformneatkarīgs dokumentu formāts. *De facto* standarts elektronisku dokumentu izplatīšanā Internetā.

RegEx – *Regular Expression* – veids kā programmai norādīt šablonu (regulāru izteiksmi), kuru meklēt tekstā.

SAX – *Simple API for XML* – uz notikumiem balstīts API, kas ļauj analizēt XML dokuments. Vienkāršota DOM alternatīva.

SGML – *Standard Generalized Mark-up Language* – vispārināta, standarta datu marķēšanas valoda (ISO 8879:1986). *De facto* standarts valodas korpusu marķēšanā. SGML izmanto par pamatu arī atvasinātu marķēšanas valodu definēšanā (piem., XML).

Servlet – *Java* platformas tehnoloģija Interneta aplikāciju izstrādei [Sun-J2EE].

SQL – *Structured Query Language* – strukturēta vaicājumu valoda (ISO 9075:1992), datu definēšanas un manipulāciju operāciju veikšanai SQL datubāzēs.

TEI – *Text Encoding Initiative* – konsorcijs, kas nodarbojas ar XML savietojamu tekstu saturiskas marķēšanas un apmaiņas formātu standartizēšanu [`http://www.tei-c.org`].

URI – *Uniform Resource Identifier* – simbolu virkne (adrese), kas identificē resursu Internetā. URL (*Uniform Resource Locator*) ir neformāls URI sinonīms.

Valodas korpuss – liels, mašīnlasāmu (*machine-readable*) tekstu masīvs. Balansēts valodas korpuss – korpuss, kas maksimāli reprezentē šīs valodas funkcionālo stilu daudzveidību [McEnery, Spektors].

W3C – *World Wide Web Consortium* – Interneta Konsorcijs. Nodarbojas ar Interneta izstrādes tehnoloģiju un specifikāciju izveidi [`www.w3.org`].

XML – *eXtensible Mark-up Language* – paplašināma marķēšanas valoda, SGML apakškopa [W3C-XML]. Tā ir formāla valoda informācijas loģiskās struktūras aprakstīšanai un datu apmaiņai (piem., Internetā).

XPath – *XML Path Language* – XML dokumentu elementu adresācijas valoda, izveidota lietošanai ar XSLT un *XPointer* tehnoloģijām [W3C-XPath].

XPointer – *XML Pointer Language* – valoda, kas paredzēta datu fragmentu identificēšanai jebkurai URI atsaucei, kas norāda uz resursu, kura Interneta medija tips ir: `text/xml`, `application/xml`, `text/xml-external-parsed-entity`, vai `application/xml-external-parsed-entity` [W3C-XPointer].

XSL – *eXtensible Stylesheet Language* – stila lapu definēšanas valoda, kas sastāv no trīs daļām: XSLT (*XSL for Transformations*) – valoda XML dokumentu transformēšanai, *XPath* un XSL FO (*Formating Objects*) – XML vārdnīca formatēšanas semantikas specificēšanai [W3C-XSL].

BIBLIOGRĀFIJA

1. DCMI (1999), *Dublin Core Metadata Element Set*, Version 1.1, `http://dublincore.org/documents/dces`
2. Mason, Oliver (2000), *Programming for Corpus Linguistics: How to Do Text Analysis with Java*, Edinburgh Textbooks in Empirical Liguistics, Edinburgh University Press
3. McEnery, Tony & Wilson, Andrew (2001), *Corpus Linguistics*, Second Edition, Edinburgh Textbooks in Empirical Liguistics, Edinburgh University Press
4. Milčonoka, Everita (2002), *Обзор ресурсов латышского языка в Институте математики и информатики Латвийского университета*, (iesniegts publicēšanai Sanktpēterburgas Valsts universitātē)
5. Spektors, Andrejs (2000), *Latviešu valodas datorfonda izveide*, Latvijas Zinātņu akadēmijas vēstis. Sērija A (`http://www.ailab.lv/aspekt/dfond.htm`)
6. Sun Microsystems (2001), *Java 2 Platform: API Specification*, Standard Edition, Version 1.4.0, `http://java.sun.com/j2se/1.4/docs/api`
7. Sun Microsystems (2000), *Java 2 Platform: API Specification*, Enterprise Edition, Version 1.2.1, `http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api`
8. W3C, *Document Object Model*, `http://www.w3.org/DOM`
9. W3C, *Extensible Mark-up Language*, `http://www.w3.org/TR/2000/REC-xml-20001006`
10. W3C, *Extensible Stylesheet Language*, `http://www.w3.org/Style/XSL`
11. W3C, *HTML Tidy*, `http://www.w3.org/People/Raggett/tidy`
12. W3C, *XML Path Language*, `http://www.w3.org/TR/xpath`
13. W3C, *XML Pointer Language*, `http://www.w3.org/TR/xptr`