LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

MIGRĀCIJAS VALODA SATURA PĀRVALDĪBAS SISTĒMĀM

MAĢISTRA DARBS

Autors: Andis Komarovs
Stud. apl. Nr. ak05051
Darba vadītājs: dr. habil. dat. Juris Borzovs

RĪGA 2011

ANOTĀCIJA

Šajā darbā tiek apskatīta un pētīta tīmekļa satura pārvaldības sistēmu datu migrācija, izmantojot valodas ziņā neitrālu datu formātu. Darba sākumā ir aplūkoti satura migrācijas teorētiskie aspekti, un kā neitrāls formāts tiek izvēlēta paplašināmās iezīmēšanas valoda (XML), kuru paplašinot tiek iegūta tīmekļa satura iezīmēšanas valoda (WCML).

Turpinājumā ir apkopoti trīs nozīmīgāko tīmekļa satura pārvaldības sistēmu – Wordpress, Joomla un Drupal – biznesa dati, iegūstot informāciju, ko var izmantot WCML izstrādē. Balstoties uz iegūto informāciju, ir izveidota WCML versija, kas iekļauj apskatīto satura pārvaldības sistēmu datu struktūras, paredz valodas paplašināšanas iespējas un sniedz atbalstu tipisku migrācijas problēmu risināšanā.

Atslēgvārdi: satura pārvaldības sistēma, migrācija, iezīmēšanas valoda, XML

ABSTRACT

In this paper the author discusses and researches migration of web content management system data by using language-neutral data format. At the beginning of the paper the author deals with theoretical aspects of migration, and selects extensible markup language (XML), which later is extended as web content markup language, as a neutral data format.

Then business data of three major web content management systems – Wordpress, Joomla and Drupal – are summarized, as a result obtaining information that can be used in WCML development. Based on obtained information, a WCML version is created. This version includes common content management system data structures, provides language extensibility and provides support in typical migration problem resolution.

Keywords: content management system, migration, markup language, XML

AUTOREFERĀTS

Šajā darbā autors ir:

- iepazinies ar satura migrācijas teorētiskajiem aspektiem;
- veicis informācijas izpēti par iezīmēšanas valodu lietojumiem satura pārvaldības sistēmu migrācijā;
- apskatījis iezīmēšanas valodu taksonomiju un paplašināšanas iespējas;
- apkopojis nozīmīgāko tīmekļa satura pārvaldības sistēmu biznesa datus;
- balsoties uz iegūtajiem datiem, izveidojis tīmekļa satura iezīmēšanas valodu;
- izveidojis praktisku importa piemēru Joomla satura pārvaldības sistēmā.

SATURS

APZĪMĒJUMU SARAKSTS

Saīsinājums	**Skaidrojums**
API	(*ang. Application Programming Interface*) – lietojumprogrammas interfeiss
CMS	(*ang. Content management system*) – satura pārvaldības sistēma
CSV	(*ang. Comma-separated values*) – faila formāts, kurā vērtības ir atdalītas ar komatiem
DTD	(*ang. Document Type Definition*) – dokumentu tipa definīcija
ECMS	(*ang. Enterprise Content Management System*) – uzņēmumu satura pārvaldības sistēma
HTML	(*ang. HyperText Markup Language*) – hiperteksta iezīmēšanas valoda
JSON	(*ang. JavaScript Object Notation*) – JavaScript objektu notācija, datu apmaiņas formāts
PHP	(*ang. PHP: Hypertext Preprocessor*) – PHP hiperteksta procesors, programmēšanas valoda
RELAX NG	(*ang. REgular LAnguage for XML Next Generation*) – XML shēmas valoda
RSS	(*ang. Really Simple Syndication*) – satura vienkāršā sindicēšana, tīmekļa barotņu formāts
SGML	(*ang. Standard Generalized Markup Language*) – vispārinātā marķēšanas standartvaloda
SQL	(*ang. Structured Query Language*) – strukturēta vaicājumvaloda
WCML	(*ang. Web Content Markup Language*) – tīmekļa satura iezīmēšanas valoda
WCMS	(*ang. Web content management system*) – tīmekļa satura pārvaldības sistēma
WXR	(*ang. Wordpress eXtended RSS*) – Wordpress paplašināts RSS, Wordpress specifisks datu apmaiņas formāts
WYSIWYG	(*ang. What You See Is What You Get*) – attēlu veidošanas metode displeja ekrānā, kas dod iespēju lietotājam ekrānā iegūt tādu sagatavojamās lappuses attēlu, kas pēc sava izskata atbilst tās drukātajam veidolam
XHTML	(*ang. eXtensible HyperText Markup Language*) – paplašināma hiperteksta iezīmēšanas valoda
XML	(*ang. Extensible Markup Language*) – paplašināmās iezīmēšanas valoda
XSLT	(*ang. Extensible Stylesheet Language Transformations*) – paplašināmās stila lapu valodas transformācijas, XML transformāciju valoda

IEVADS

Pēdējo divdesmit gadu laikā ir notikusi strauja interneta un interneta tehnoloģiju attīstība. Tīmekļa vietnes ir attīstījušās līdzi laikam, un vienkāršu, ar roku rediģējamu HTML failu vietā ir nākušas modernas satura pārvaldības sistēmas (CMS). Līdz ar satura pārvaldības sistēmu attīstību ir kļuvis iespējams salīdzinoši viegli pārvaldīt lielus satura apjomus. Tomēr pienāk brīdis, kad pat vismodernākā satura pārvaldības sistēma ir novecojusi, un nespēj apmierināt vietnes vajadzības. Šajā brīdī ir iespējams vai nu uzlabot esošo CMS, vai pāriet uz citu, modernāku satura pārvaldības sistēmu. Pārejas gadījumā kļūst aktuāls satura migrācijas jautājums.

Satura migrācija nav pats aizraujošākais un interesantākais darbs pasaulē, tomēr kādam tas ir jāveic. Ja vietne ir neliela, tad satura migrāciju var veikt manuāli, bet ja datu apjoms ir liels, tad manuāla satura migrācija nav risinājums, jo tas ir ilgs, vienmuļš un dārgs process. Eksistē dažādi veidi, kā automatizēt satura migrāciju, un viens no šiem veidiem ir valodas ziņā neitrāla formāta izmantošana, kas izpaužas kā satura eksports no vecās sistēmas uz neitrālu formātu, un imports no šī formāta jaunajā sistēmā.

Ja satura migrācija ir jāveic tikai starp divām sistēmām, tad situācija nav tik slikta, jo eksistē CMS specifiski migrācijas formāti, un ja abām sistēmām ir kopīgs atbalsts kādam no specifiskajiem formātiem, tad satura migrācija, ideālā gadījumā, var būt dažu klikšķu jautājums.

Situācija ir sarežģītāka, ja ir iesaistītas vairākas satura pārvaldības sistēmas. Šāda situācija var gadīties, ja ir jāveic satura migrācija no vairākām sistēmām. Jo vairāk dažādu sistēmu, jo mazākas cerības, ka tās visas atbalstīs vienu formātu. Ja eksistētu vienots satura migrācijas formāta standarts, CMS atliktu pievienot tikai šī standarta atbalstu, un būtu iespējams migrēt datus no, vai uz jebkuru citu satura pārvaldības sistēmu. Diemžēl šāds standarts neeksistē, šķiet vistuvāk tam ir satura sindicēšanas formāti, no kuriem populārākie ir RSS [1] un Atom [2], tomēr arī tie ir paredzēti ziņu apkopošanai, nevis to migrācijai.

Šī darba mērķis ir izveidot šāda vienota satura formāta skici. Kā realizācijas tehnoloģija ir izvēlēta paplašināmās iezīmēšanas valoda (XML), uz kuras bāzes tiks veidota tīmekļa satura iezīmēšanas valoda (WCML – *Web Content Markup Language*). Lai varētu izpildīt mērķi, vispirms ir jāizpēta satura migrācijas process, tad jānoskaidro biežāk sastopamo satura pārvaldības datu struktūra, un galu galā, jāizveido WCML specifikācija.

Darba pirmajā nodaļā tiek apskatīti vispārīgi satura migrācijas soļi, migrācijas automatizācijas iespējas un priekšnosacījumi satura automatizētai migrācijai, kā arī tiek

pievērsta pastiprināta uzmanība neitrāla datu formāta izmantošanai satura migrācijā, t.s. arī informācijas izpētei par šo tēmu.

Otrajā nodaļā tiek apskatīti satura pārvaldības sistēmu veidi, un kā izpētes objekts izvēlētas tīmekļa satura pārvaldības sistēmas. Turpinājumā tiek apskatītas nozīmīgākās tīmekļa satura pārvaldības sistēmas, un apkopota informācija par to uzglabātajiem datiem.

Trešajā nodaļā tiek apskatītas iezīmēšanas valodas, to attīstības gaita un apskatīta jaunu valodu veidošana uz esošu iezīmēšanas valodu bāzes.

Ceturtajā nodaļā ir aprakstīta WCML valoda, un šī nodaļa kalpo kā WCML specifikācija. Nodaļas beigās ir parādīts neliels, praktisks importa piemērs Joomla satura pārvaldības sistēmā.

1. SATURA MIGRĀCIJA

Satura migrācijai no biznesa skatu punkta var būt daudz un dažādi iemesli, tomēr tehniski satura migrācijai ir divi iemesli:

1) jaunas tehnoloģijas, piemēram, satura pārvaldības sistēmas izvēle;
2) esošās sistēmas pārveidošana lai samazinātu tās trūkumus vai pievienotu būtisku funkcionalitāti [3].

Šie iemesli arī nosaka, vai satura migrācija ir jāveic starp divām dažādām sistēmām, vai starp vienas sistēmas dažādām versijām. Šajā darbā, runājot par satura migrāciju, tiek domāts pirmais variants, tomēr visu minēto informāciju ir iespējams attiecināt arī uz datu migrāciju starp vienas satura pārvaldības sistēmas dažādām versijām.

Šajā nodaļā ir īsi apskatīti satura migrācijas soļi, migrācijas automatizācija, iezīmēšanas valodu lietojums un neitrāla datu formāta izmantošana satura migrācijā.

1.1. Satura migrācijas soļi

Jebkura datu migrācija sastāv vismaz no trīs soļiem – datu ieguves, transformācijas un ielādes. Izņēmums ir vienkārša datu kopēšana, jo šajā gadījumā netiek veikta datu transformācija. Citos gadījumos vispirms no datu avota ir jāiegūst nepieciešamie dati, jāpārveido tie nepieciešamajā formā, nepieciešamības gadījumā jāveic arī datu apstrāde, un tad jāielādē tie mērķa sistēmā. Atkarībā no situācijas, dati var tikt migrēti vai nu vienā piegājienā, vai arī atkārtojot šos soļus iteratīvi.

1.1.1. Datu ieguve

Pirmais no minētajiem datu migrācijas soļiem ir datu ieguve. Datu ieguves primārais mērķis ir iegūt un pārveidot datus no datu avota vai avotiem tādā formā, kas ir piemērota datu transformāciju veikšanai. Tādēļ datu formāta maiņa, ja vien tās laikā netiek mainīta datu struktūra, ir datu ieguves procesa, nevis datu transformācijas, sastāvdaļa.

Tīmekļa vietņu migrācijā iegūstamos datus var sadalīt šādās grupās:

1) **tekstuālais saturs** – šajā grupā ietilpst visi dati, kas tiek attēloti lapas HTML kodā, t.s. arī HTML metadati;
2) **binārais saturs** – lapas attēli, pielikumi un cita veida binārie faili;
3) **saites** – lapā sastopamās hipersaites;
4) **struktūra** – kategorijas, tagi, izvēlnes, u.tml.

Datu ieguves sarežģītība ir atkarīga no izmantotajām tehnoloģijām, ja tiek izmantota satura pārvaldības sistēma, tad, iespējams, datus var izeksportēt automātiski, vai veikt datu apstrādi uzreiz datubāzē, savukārt, ja vietne sastāv no nestrukturētiem HTML failiem, tad datu ieguvei, iespējams, nāksies izmantot rāpuli, kas apstaigā doto vietni, un no lapas koda iegūst nepieciešamo informāciju.

Datu ieguvē var nākties saskarties ar dažādām problēmām, par kurām sākotnēji var būt grūti iedomāties, tādēļ gadās, ka šīs problēmas tiek pamanītas tikai tad, kad datu migrācija rit pilnā sparā, nevis jau plānošanas fāzē. Lūk dažas no šīm problēmām [4]:

Nepieejams saturs – digitālais saturs var būt nepieejams dažādu iemeslu dēļ. Faili, raksti vai saturs, kas atrodas internetā, piemēram, YouTube un Flickr vietnēs, var tikt izdzēsts vai pārvietots, tādā veidā radot bojātas saites.

Dinamiskas lapas un resursi – jau statiska satura migrēšana var būt sarežģīta, bet dinamiska satura migrēšana sarežģītību var pacelt jaunā līmenī. Web 2.0 tehnikas, piemēram, Javascript, kas dinamiski maina lapas renderēšanas laikā var padarīt nelietojamas migrācijas metodes, kas paļaujas uz statiskām lapām. Šī pati problēma attiecas arī uz datu ieguvi no satura pārvaldības sistēmām. Piemēram, Joomla satura pārvaldības sistēmā ir spraudņi, kas spēj mainīt raksta saturu pēc tā atlases no datubāzes – šādā gadījumā no datubāzes atlasītā raksta saturs nesakritīs ar vietnē parādītā raksta saturu.

Metadatu ieguve un migrācija – tīmekļa lapu metadatiem ir liela nozīme meklēšanas dzinēju optimizācijā, un ja tīmekļa vietnei ir svarīga tās pozīcija meklēšanas rezultātos, tad ir ļoti vēlams, lai lapai būtu metadati. Ir labi, ja esošajai vietnei jau eksistē nepieciešamie metadati, bet ja tā nav, tad var nākties nopūlēties, lai šos datus iegūtu.

Saišu ieguve – tīmekļa vietnes strādā tādēļ, ka eksistē saites starp dažādiem resursiem – gan lapām, gan attēliem un failiem, tomēr saišu saglabāšana starp resursiem nav triviāls uzdevums. Jaunajā vietnē katram resursam ir savādāka adrese, tādēļ vispirms ir jāiegūst visas vecās saites no eksistējošās vietnes, tās jāpārnes uz jauno vietni, un tad vēlreiz jāapstaigā visas vecās lapas, lai atjaunotu saites. Šo procesu vēl sarežģītāku padara meklēšanas dzinējiem pielāgotās saites, jo no tām nevar tiešā veidā iegūt satura identifikatoru datubāzē.

1.1.2. Datu transformācija

Pēc datu ielādes iegūtie dati ir jāpārveido tādā formā, kas ir piemērota datu ielādei jaunajā sistēmā. Šis solis ir datu transformācija, kas sevī ietver datu kartēšanu starp avota un mērķa datu elementiem, lai izveidotu atbilstības starp abu sistēmu datiem. Lūk daži datu transformācijas laikā veicamo darbību piemēri:

- kodēto vērtību tulkošana (piem., „Vīr.” pārveido uz 1, „Siev.” uz 2)

- atvasināto vērtību aprēķināšana (piem., summa = cena * daudzums)
- lauku filtrēšana (piem., ignorē laukus, kuri nav nepieciešami)
- datu kārtošana
- datu pagriešana (piem., datubāzes kolonnu pārveidošana par rindām, un otrādi)
- apvienošana un šķelšana (piem., teksta virknes sadalīšana pa vairākiem laukiem vai apvienošana no vairākiem laukiem)
- utt.

Dažbrīd datu transformācija var šķietami saplūst ar datu ieguvi un ielādi, piemēram, no satura pārvaldības sistēmas tiek izeksportēts XML fails, kurš savukārt tiek ielādēts citā satura pārvaldības sistēmā. Šajā procesā datu ieguve ir XML eksports, bet ielāde – XML imports, savukārt datu transformācija it kā pazūd. Šajā piemērā ir jāizšķir divi gadījumi:

1) dati tiek migrēti starp vienādām sistēmām;
2) dati tiek migrēti starp dažādām sistēmām.

Pirmajā gadījumā tiešām datu transformācija var praktiski izpalikt, jo datu struktūras ir vienādas, un tādēļ faktiski notiek datu kopēšana. Otrajā gadījumā kādā brīdī tomēr ir jāveic datu transformācija, jo, visticamāk, atšķiras veids, kā tiek uzglabāti dati. Dotajā piemērā datu transformācija var notikt vai nu eksportējot datus uz XML, vai nu importējot tos no XML faila, vai arī abu procesu laikā (ja XML struktūra neatbilst nevienai no sistēmām).

1.1.3. Datu ielāde

Pēc datu transformācijas ir nepieciešama datu ielāde jaunajā sistēmā. Sākumā var šķist, ka šī ir ļoti vienkārša migrācijas daļa, jo datu transformācijas laikā jau ir sagatavoti dati, kurus tikai atliek ielādēt. Tā tas tiešām var būt, ja CMS piedāvā datu importa funkcionalitāti. Tomēr šī funkcionalitāte var būt nepilnīga, vai tās var nebūt vispār. Un pat ņemot vērā to, ka populārākās satura pārvaldības sistēmas glabā datus labi strukturētās datubāzēs, datu ielādē ir jāsaskaras ar dažādām problēmām.

Pirmkārt tas ir datu validācijas jautājums. Ja CMS nepiedāvā iebūvētas funkcijas datu importam, tad atliek divi risinājumi:

1) dublēt datu validācijas pārbaudes (kuras ne vienmēr ir triviālas pārbaudes par to vai ir aizpildīts obligātais lauks);
2) izveidot robotu, kas simulē cilvēka darbību un ievada datus caur satura pārvaldības sistēmas saskarni.

Otrkārt tā ir atkarība no esošajiem datiem. Tie var būt gan dažādi dati, kurus nevar iegūt transformācijas laikā, gan dažādi algoritmi, kurus var realizēt tikai datu ielādes laikā. Viens

no piemēriem ir iekļauto kopu modelis (*nested set model*), kuru izmanto hierarhisku datu attēlošanai datubāzu tabulās [5]. Šis modelis izmanto divas kolonnas *left* un *right*, lai uzturētu hierarhiju starp tabulas rindām. Ja tabulā, kurā tiek izmantots šis modelis, ievieto vienu rindu, tad ir jāatjauno arī pārējās rindas, lai vērtības *left* un *right* kolonnās veidotu korektu hierarhisko modeli.

Treškārt tās ir saites, šis gadījums pēc būtības ir jau minētā atkarība no datiem, tikai tas ir izdalīts atsevišķi, jo ar šo problēmu ir jāsaskaras jebkurā tīmekļa satura migrācijā. Lai pārtulkotu saites, ir nepieciešams vispirms ieimportēt rakstus jaunajā CMS, jo, lai izveidotu saiti, ir jāzina tā raksta identifikators, uz kuru norāda šī saite. Turklāt situācija kļūst tikai sarežģītāka, ja saite norāda nevis uz rakstu, bet gan kādu citu resursu, piemēram, kalendāru.

1.2. Satura migrācijas automatizācija

Satura migrācija, ja to veic manuāli, ir ļoti nogurdinošs, vienveidīgs un laikietilpīgs darbs, tādēļ ir saprotama vēlme šo procesu automatizēt, tomēr tas ne vienmēr ir iespējams. Praksē ir trīs galvenie satura migrācijas veidi [3]:

1) manuāla migrācija;
2) daļēji automatizēta migrācija;
3) automatizēta migrācija.

Katram no šiem veidiem ir savas stiprās un vājās puses, un tas, kuru no šiem veidiem izvēlēties ir atkarīgs no situācijas, tādēļ nevienu variantu nevar viennozīmīgi ieteikt vai ignorēt.

1.2.1. Manuāla migrācija

Manuālā migrācija ir visvienkāršākā, bet arī visdarbietilpīgākā metode, kas arī ir tās lielākais trūkums. Praksē šī metode izpaužas kā satura kopēšana no vecās vietnes jaunajās vietnes administrācijas rīkā.

Manuālās migrācijas priekšrocība ir tās elastība, kas ļauj pārstrukturēt un attīrīt saturu tā, kā tas nav iespējams ar citām metodēm. Tomēr šo darbu vajadzētu uzticēt cilvēkiem, kas saprot, kādai jābūt jaunās vietnes struktūrai, pārzina esošo struktūru, un rezultātā spēs uzlabot satura kvalitāti.

Manuāla migrācija ir iespējama, ja lapu skaits nav pārāk liels, tipiski zem 5000 [6], jo citādi darba apjoms ir pārāk liels.

1.2.2. Automatizēta migrācija

Vispievilcīgākais migrācijas veids ir automatizēta satura migrācija. Šajā gadījumā praktiski nav nepieciešama vai nepieciešama minimāla manuāla iejaukšanās.

Ir vairāki veidi, kā veikt automatizētu datu migrāciju:

1) izmantojot CMS pieejamās importa/eksporta iespējas, izeksportēt datus no vecās sistēmas kādā valodas ziņā neitrālā formātā un tad ieimportēt šos datus jaunajā sistēmā;
2) izmantojot piedāvātās API funkcijas, izveidot programmu, kas pārnes saturu starp sistēmām;
3) izmantojot reverso inženieriju, iegūt datus no vecās CMS datubāzes, pārveidot tos nepieciešamajā formātā, un tad ieimportēt jaunajā sistēmā;
4) izmantojot robotu apstaigāt veco vietni un izvilkt datus no HTML koda, pēc tam iegūtos datus ieimportēt jaunajā sistēmā;
5) izmantot trešās puses migrācijas risinājumu, kas piedāvā rīkus un tehnikas automatizētai migrācijai (tomēr iekšēji šie rīki tik un tā izmanto augstāk minētos veidus).

Visvienkāršākais ir pirmais variants, jo tas neprasa manuālu iejaukšanos, piemēram, Wordpress satura pārvaldības sistēmā ir iespēja izeksportēt datus WXR (*Wordpress eXtended RSS*) formātā, savukārt Drupal satura pārvaldības sistēmai eksistē modulis [7], kurš atbalsta šī formāta ielādi. Tomēr arī šāda pieeja nav universāla, un gadījumos, ja importa un eksporta formāti nesakrīt, tomēr ir nepieciešams izveidot transformāciju starp šiem formātiem.

Neatkarīgi no tā, kurš no šiem veidiem ir izvēlēts, ir vairāki priekšnosacījumi, lai varētu veikt automatizētu satura migrāciju:

- esošajā vietnē ir jābūt augstas kvalitātes saturam, citādi migrācija pārvēršas par „atkritumu" pārnešanu starp sistēmām;
- jaunās vietnes struktūrai ir jābūt līdzīgai vecās vietnes struktūrai, citādi transformācijas likumi kļūst pārāk sarežģīti;
- vecās vietnes HTML kodam jābūt pietiekoši "tīram" un konsekventam, un tas attiecas ne tikai uz vietnes apstaigāšanu ar robotu, bet arī uz citiem migrācijas veidiem, jo, piemēram, rakstu saturs, visbiežāk, tiek glabāts HTML formātā.

Ja neizpildās šie priekšnosacījumi, tad pilnībā automatizēt satura migrāciju nav iespējams.

1.2.3. Daļēji automatizēta migrācija

Pat ja vecās vietnes struktūra un kvalitāte ir pārāk sliktas, lai veiktu pilnīgi automatizētu satura migrāciju, dažas vietnes daļas var būt strukturētas labāk, un tādēļ piemērotas automatizētai datu migrācijai. Piemēram, vietnē ir vairāki simti preses paziņojumu, kuri ir ieturēti vienotā stilā. Automātiski ieimportējot šos preses paziņojumus var krietni vien ietaupīt laiku un izmaksas.

Cits veids, kā daļēji automatizēt migrāciju ir skriptu veidošana veicamajiem darbiem. Piemēram, jaunajā sistēmā visi attēli ir jāievieto mapē „images". Ja šo darbu veic manuāli, tad visi faili ar paplašinājumu .jpg un .png ir jāpārvieto uz „images" mapi, turklāt ir jāpārskata visi raksti, un jāizlabo saites uz attēliem. Šis ir vienmuļš, un garlaicīgs, toties labi automatizējams darbs, ko var uzticēt skriptam.

1.3. Neitrāla datu apmaiņas formāta izmantošana satura migrācijā

Kā minēts iepriekš, viens no veidiem, kā automatizēt satura migrāciju ir valodas ziņā neitrāla formāta izmantošana, lai pārnestu datus starp sistēmām. Šādas migrācijas piemērs parādīts attēlā 1.1.

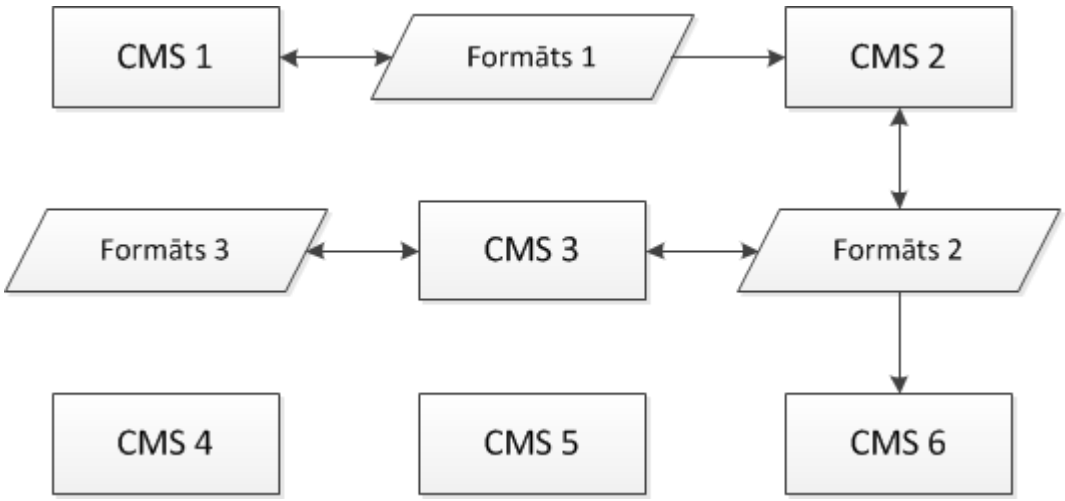
1.1. att. **Neitrāla formāta izmantošana starp divām satura pārvaldības sistēmām**

Lai varētu veikt datu migrāciju šādā veidā, ir nepieciešams, pirmkārt, eksporta atbalsts sistēmā, no kuras iegūst datus, un, otrkārt, importa atbalsts sistēmā, kurā ir jāielādē dati. Šis atbalsts var būt vai nu iekļauts sistēmā, vai pievienojams ar paplašinājumu palīdzību. Savukārt, ja šāda atbalsta nav, situācija ir sarežģītāka, jo ir jāveido pašam savs paplašinājums, toties pēc šāda paplašinājuma izveides to ir iespējams atkārtoti izmantot citā projektā.

Šāds datu migrācijas veids ir pietiekoši ērts, jo praktiski nav nepieciešama manuāla iejaukšanās. Ideālā gadījumā vienā CMS, nospiežot pogu eksportēt, iegūst failu ar datiem nepieciešamajā formātā, un otrā CMS, nospiežot pogu importēt, šie dati tiek ieimportēti. Tomēr, šādā veidā var migrēt tikai tos datus, kas ir paredzēti formāta specifikācijā, bet ja ir nepieciešams migrēt citus datus, tad nākas vai nu paplašināt esošo formātu un pievienot paplašinājumu atbalstu eksporta un importa procedūrām, vai arī atlikušos datus migrēt citā veidā.

1.3.1. Migrācija starp vairākām sistēmām

Praksē var gadīties situācijas, kad ir nepieciešams atbalstīt satura migrāciju uz vai no vairākām sistēmām. Piemēram, satura pārvaldības sistēmā nepieciešams ieimportēt datus no divām vai vairāk sistēmām, vai arī CMS izstrādātāji vēlas atbalstīt datu importu no vairākām sistēmām, lai lietotājiem būtu vieglāk pāriet uz viņu sistēmu. Situāciju sarežģī tas, ka neeksistē plaši akceptēts standarta formāts satura apmaiņai starp satura pārvaldības sistēmām (sīkāk skat. 1.4 apakšnodaļu). Attēlā 1.2 ir parādītas dažādas kombinācijas starp sistēmām, kas ir iespējamas bez vienota satura apmaiņas standarta.



1.2. att. **Satura apmaiņa starp CMS bez vienota datu apmaiņas standarta**

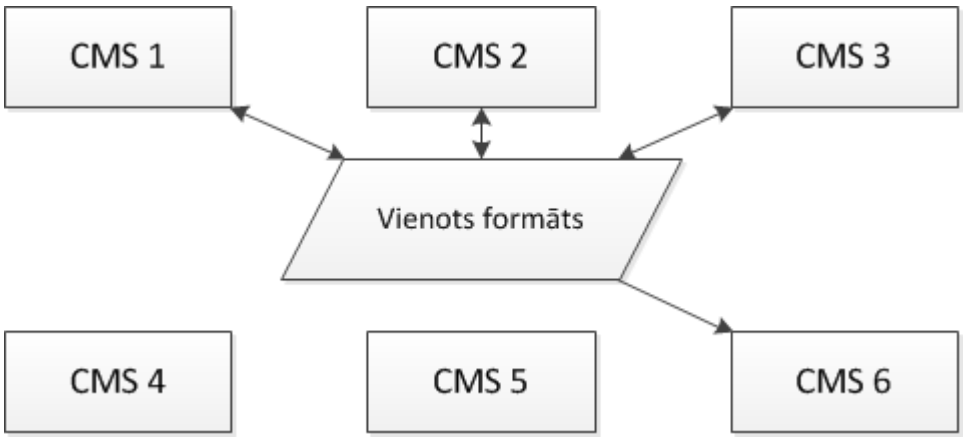
Kā redzams, dažas CMS atbalsta tikai vienu datu apmaiņas formātu, dažas – vairākus, bet dažas neatbalsta vispār nevienu, tāpat atšķiras tas, vai tiek atbalstīts gan eksports, gan imports, vai arī tikai imports. Ja šādā situācijā ir jāmigrē dati no vienas CMS uz otru, tad no migrācijas viedokļa ir iespējamas šādas situācijas:

1) pilns atbalsts – eksistē kopīgs formāts, uz kuru no avota CMS var izeksportēt datus, un no kura mērķa sistēmā var ieimportēt datus (piemēram, 1.2. attēlā CMS 1 → CMS 2, bet ne CMS 6 → CMS 2);
2) daļējs atbalsts – viena vai abas sistēmas atbalsta kādu datu eksporta vai importa formātu, bet tas nav kopīgs (CMS 1 → CMS 3, CMS 3 → CMS 4) , vai arī datu migrācija pa to iespējama tikai pretējā virzienā (CMS 6 → CMS 2);
3) nav atbalsta – nevienai no sistēmām nav importa vai eksporta atbalsta (CMS 4 → CMS 5).

Pirmais gadījums īpašus komentārus neprasa, savukārt otrajā un trešajā gadījumā ir nepieciešama pielāgota izstrāde, kuras laikā vai nu izveido transformāciju starp dažādiem datu apmaiņas formātiem, vai arī nepieciešamajai CMS pievieno kāda formāta atbalstu. Turklāt

otrais un trešais gadījums atšķiras arī ar to, ka otrajā gadījumā jau datu apmaiņas formāts lielākoties ir gatavs, un nepieciešamības gadījumā tikai jāpaplašina, bet trešajā gadījumā ir vai nu jāveido savs formāts, vai jāizvēlas kāds vispārīgs formāts, un tas jāpaplašina.

Attēlā 1.3 ir parādīts, kas notiktu, ja 1.2 attēlā redzamos trīs formātus apvienotu vienā, un satura migrācijai izmantotu vienotu datu apmaiņas formātu. Ir redzams, ka pēc apvienošanas starp CMS 1, CMS 2 un CMS 3 ir iespējama satura migrācija jebkurā virzienā, tāpat no šīm trijām sistēmām ir iespējams migrēt datus uz CMS 6. Tāpat kļūst skaidrāka rīcība CMS 4 → CMS 5 migrācijas gadījumā, jo ir izdevīgāk nevis veidot savu formātu datu apmaiņai, bet gan izmantot vienoto formātu, jo tādejādi, nepieciešamības gadījumā, tiek nodrošināta arī datu migrācija uz citām sistēmām.



1.3. att. **Satura apmaiņa starp CMS, izmantojot vienotu datu apmaiņas formātu**

Vienota formāta priekšrocības ir acīmredzamas, tomēr šajā gadījumā ir nepieciešams līdzsvars starp pārāk universālu un pārāk CMS specifisku formātu. Vienā šī diapazona galā atrodas vispārīgs formāts, kurš nosaka tikai formāta uzbūvi, bet ne struktūru, piemēram, XML, savukārt otrā galā atrodas CMS specifisks formāts, kas, praktiski bez datu zudumiem, ļauj eksportēt un importēt saturu no dotās satura pārvaldības sistēmas. Jo universālāks ir formāts, jo vairāk nepieciešams to pielāgot katram gadījumam, savukārt pārāk specifiskiem formātiem ir ierobežots pielietojums.

1.3.2. Divvirzienu migrācija

Apskatot 1.3 attēlu, rodas loģisks jautājums – vai vienoto datu apmaiņas formātu ir iespējams izmantot abos virzienos, t.i. datu sinhronizācijā? Atbilde ir – iespējams, jā, bet tas, visticamāk, prasīs eksporta/importa procedūru pielāgojumus. Turklāt jāņem vērā tas, ka jo universālāks būs datu apmaiņas formāts, jo vairāk pūļu būs jāpieliek, lai nezaudētu datus sinhronizācijas rezultātā. Šajā darbā šis jautājums darbā netiek pētīts, jo tas ir ārpus darba

tēmas, turklāt saistās ar virkni potenciālu problēmu, tādu kā konfliktu risināšana, ierakstu identificēšana, datu zudumi, u. tml.

1.3.3. Formāta izvēle

Vienota datu migrācijas formāta izveidē ir svarīgs arī tā tehniskais pamats. Satura pārvaldības sistēmas tiek veidotas izmantojot gan dažādas programmēšanas valodas, gan dažādas datubāzu pārvaldības sistēmas, tādēļ ir svarīgi, lai datu apmaiņā tiktu izmantots valodas ziņā neitrāls formāts. Šāds formāts var būt gan teksta, gan binārs fails, bet lai to spētu korekti nolasīt abas datu apmaiņā iesaistītās sistēmas, tas nedrīkst saturēt kādai valodai specifisku saturu. Piemēram, CSV (*comma separated value*) fails ir valodas ziņā neitrāls formāts, savukārt teksta fails, kas satur serializētu PHP objektu tāds nav.

XML un JSON ir divi izplatīti un valodas ziņā neitrāli formāti, un katram no tiem ir savas priekšrocības un trūkumi [8]. Tie abi ir teksta formāti, turklāt tos ir iespējams konvertēt no viena formāta otrā [9], tomēr starp tiem ir kāda atšķirība – JSON ir datu apmaiņas formāts, bet XML - dokumentu iezīmēšanas valoda. Tādēļ JSON ir krietni vienkāršāks formāts, kas labāk piemērots datu struktūru transportēšanai no viena punkta uz otru, savukārt XML ir vairāk piemērots dokumentu uzglabāšanai un pārsūtīšanai starp vairākām sistēmām, turklāt tas ir paplašināms un atbalsta sarežģītāku struktūru veidošanu [10].

Minēto iemeslu dēļ, šajā darbā kā datu apmaiņas formāta pamats ir izvēlēts XML.

1.4. Esošie satura apmaiņas standarti un formāti

Pirms veidot jaunu datu apmaiņas formātu, ir jāapskata jau esošie formāti un jāsameklē informācija par šo tēmu, lai noskaidrotu vai jau neeksistē kāds ļoti līdzīgs vai pat tieši tāds pats risinājums. Informācijas meklēšanā tika izmantoti šādi resursi:

- ScienceDirect (http://www.sciencedirect.com/)
- SpringerLink (http://www.springerlink.com/)
- ebrary (http://www.ebrary.com/)
- Google Scholar (http://scholar.google.com/)
- Google (http://www.google.com/)

Darba tēma ir saistīta ar satura pārvaldības sistēmām, satura migrāciju starp tām un iezīmēšanas valodas veidošanu, turklāt darba 2. nodaļā tiek sīkāk apskatītas tīmekļa satura pārvaldības sistēmas, tai skaitā Drupal, Joomla un Wordpress.

Ņemot vērā šos datus, sākotnējai informācijas meklēšanai norādītajos resursos tika izmantoti:

- atslēgvārdi – cms migration standard, content migration markup language, content management system, web content, drupal, joomla, wordpress, migration, export, import, data exchange;
- atslēgvārdu kombinācijas – content management system migration, content management system export, content management system import, content management system data exchange, web content migration, web content export, web content import, web content data exchange, drupal migration, drupal export, drupal import, drupal data exchange, joomla migration, joomla export, joomla import, joomla data exchange, wordpress migration, wordpress export, wordpress import, wordpress data exchange;
- informācija un avoti no atrastajiem resursiem.

Turpinājumā ir īsi apskatīti daži esošie tīmekļa saturu pārvaldības sistēmu datu apmaiņas formāti.

RSS (*Really Simple Syndication*) [1] ir vienkāršs tīmekļa satura sindicēšanas formāts, kas paredzēts informācijas apkopošanai standartizētā veidā. RSS ļauj plūsmu lasītājiem (*feed reader*) automātiski apkopot jaunākos ierakstus no daudzām vietnēm, bez nepieciešamības tās apmeklēt. Tehniski RSS ir XML formāts, kas iekļauj pilnu rakstu vai tā ievadu, un dažādus metadatus, tādus kā publicēšanas datums un autors.

Atom [2] ir cits, XML bāzēts, tīmekļa satura sindicēšanas formāts. Tam ir mazliet vairāk iespēju nekā RSS, piemēram, ir iespējams norādīt teksta formātu (teksts vai HTML), vai atsevišķi izdalīt ievadu un pamattekstu, kas RSS tiek norādīts vienā elementā.

WXR (*Wordpress eXtended RSS*) ir Wordpress satura pārvaldības sistēmai specifisks eksporta un importa formāts [11]. Tehniski tas ir veidots uz RSS bāzes un ir tā paplašinājums. Šādu, CMS specifisku formātu ir daudz, bet šis ir pieminēts, kā paraugs, jo tas ir specifisks vienai no darbā apskatītajām satura pārvaldības sistēmām.

CMIS (*Content Management Interoperability Services*) [12] standarts definē domēna modeli un tīmekļa pakalpes, un Restful AtomPub saistījumus (*bindings*), ko var izmantot lietojumprogrammatūras, lai sadarbotos ar vienu vai vairākām satura pārvaldības sistēmām. Tajā tiek izmantots arī iepriekš pieminētais Atom formāts. Šis standarts ir paredzēts lai uzlabotu savstarpējo sadarbību starp uzņēmumu satura pārvaldības sistēmām, un tā modelis ir balstīts uz kopēju dokumentu pārvaldības sistēmu arhitektūru. Lai gan CMIS piedāvā datu modeli, tas ir API standarts, nevis datu apmaiņas formāts.

Veicot informācijas izpēti ir secināts, ka neeksistē vienots datu apmaiņas formāta standarts satura pārvaldības sistēmām, un esošie formāti ir vai nu vienkārši un vispārīgi, vai arī paredzēti konkrētai satura pārvaldības sistēmai.

2. SATURA PĀRVALDĪBAS SISTĒMAS

Šajā darbā tiek veidota domēna specifiska iezīmēšanas valoda WCML, kas kalpo arī kā datu apmaiņas formāts, un kā domēns šai valodai ir izvēlētas satura pārvaldības sistēmas. Katrai satura pārvaldības sistēmai ir noteikta struktūra un datu lauki, kas tajā tiek uzglabāti, tāpat arī iezīmēšanas valoda sastāv no noteiktas struktūras un laukiem. Tādēļ, lai izveidotu WCML, ir nepieciešams noteikt, kādi būs šīs valodas datu lauki, un to pēc būtības ir iespējams izdarīt divos veidos – izveidojot CMS lauku kopas šķēlumu vai apvienojumu. Šis ir ļoti vienkāršots skatījums, tomēr tādā veidā var ilustrēt problēmu, kas rodas izvēloties pārāk plašu domēnu. Piemēram, eksistē satura pārvaldības sistēma dokumentiem - X, un satura pārvaldības sistēma uzņēmuma kontaktiem Y. Šīm sistēmām, visticamāk ir, diezgan maz kopīgu datu lauku, tādēļ izvēloties datu apmaiņas formātam tikai kopīgos laukus, to sanāks ļoti maz, savukārt apvienojot visus X un Y laukus, to sanāks ļoti daudz. Tālāk pieņemsim, ka laika gaitā X un Y sistēmas ir paplašinātas, un to funkcionalitāte ir kļuvusi pietiekoši līdzīga, tādēļ ir nepieciešama datu migrācija no vienas sistēmas uz otru. Ja datu apmaiņas formātā būs pārāk maz lauku, migrācijas projektā būs nepieciešama papildus izstrāde, lai pārnestu tos datus, kas nav iekļauti šajā formātā. Savukārt, ja būs iekļauti visi lauki, tad papildus izstrāde nebūs nepieciešama. Ja formātam ir jāatbalsta tikai divas sistēmas, tad otrais risinājums ir pieņemams, bet ja šos sistēmu ir daudz, daudz ir arī datu lauku, kas galu galā datu apmaiņas formātu padarītu milzīgu un grūti realizējamu. Tieši tādēļ darbā tiek apskatītas nevis visas satura pārvaldības sistēmas, bet gan tikai tīmekļa satura pārvaldības sistēmas, tomēr darbā vienkāršības labad, tās tiek sauktas vienkārši par satura pārvaldības sistēmām, ja vien no konteksta nav saprotams, ka tiek runāts par visu satura pārvaldības sistēmu kopu.

Šīs nodaļas sākumā ir sniegts ieskats satura pārvaldības sistēmu veidos un īsi aprakstītas nozīmīgākās tīmekļa satura pārvaldības sistēmas, savukārt turpinājumā ir apkopota informācija par šo sistēmu uzglabātajiem biznesa datiem. Šī informācija turpmāk kalpos kā pamats tīmekļa satura iezīmēšanas valodai.

2.1. Satura pārvaldības sistēmu veidi

Saturs ir ļoti plašs jēdziens, un tam var būt dažādas nozīmes, bet satura pārvaldības sistēmu kontekstā ar saturu saprot digitālo informāciju, kas tiek glabāta pārvaldības sistēmā. Saturs var būt gan raksts kādā vietnē, gan attēls, video vai mūzikas fails, gan, piemēram, e-pasta ziņojums. Daudzas mūsdienu satura pārvaldības sistēmas spēj darboties ar visiem šiem satura veidiem, turklāt to iespējas var palielināt pievienojot dažādus paplašinājumus, kas

apgrūtina sistēmu klasifikāciju. Tomēr, lai gan robeža starp sistēmām ir izplūdusi, tās ir iespējams klasificēt pēc pielietojuma veida.

Tīmekļa satura pārvaldības sistēmas ir paredzētas tīmekļa vietņu pārvaldībai, un ļauj lietotājam bez tehniskām zināšanām viegli pievienot un strukturēt vietnes saturu, izveidot un uzturēt navigācijas izvēlnes, kā arī mainīt vietnes izskatu, izmantojot dažādas veidnes. Šis CMS veids ir tas, ko parasti iedomājas un saprot ar nosaukumu satura pārvaldības sistēma. Piemērs – Wordpress (http://wordpress.org/).

Uzņēmumu satura pārvaldības sistēmas ir veidotas tieši uzņēmumu un organizāciju vajadzībām, un nepieciešamības gadījumā var iekļaut tīmekļa satura pārvaldību, tomēr neaprobežojas ar to. Šī veida sistēmas, atkarībā no uzņēmuma prasībām, var iekļaut lietotāju, dokumentu, e-pastu, notikumu, digitālā īpašuma un cita veida informācijas pārvaldību, glabāšanu un uzturēšanu. Piemērs – Alfresco (http://www.alfresco.com/).

Mobilā satura pārvaldības sistēmas ir paredzētas satura un pakalpojumu piegādei mobilajiem tālruņiem, viedtālruņiem un personālajiem ciparasistentiem, un aktīvi izmanto dažādas veidnes, lai pielāgotos mobilo ierīču daudzveidībai. Šīs sistēmas var būt gan veidotas atsevišķi, gan kā citas satura pārvaldības sistēmas sastāvdaļa. Mobilā satura pārvaldības sistēmām ir jāspēj atpazīt un pielāgoties ierīču prasībām un citiem ierobežojumiem, piemēram, ekrāna izmēram, savienojuma ātrumam, u. tml. Tāpat šīs satura pārvaldības sistēmas var ņemt vērā ierīces atrašanās vietu, izmantot to dažādu pakalpojumu piedāvāšanai vai ierobežošanai. Piemērs – Synapsy (http://www.synapsy.com/).

Komponentu satura pārvaldības sistēmas darbojas ar saturu komponentu, nevis dokumentu līmenī. Kā komponents var kalpot, gan raksts, gan tabula, attēls, rindkopa vai pat atsevišķs vārds. Katram komponentam ir savs dzīves cikls (īpašnieks, versija, lietojumi, utt.), un to mainot, tas tiek izmainīts visās vietās, kur tas ir lietots. Tādā veidā tiek nodrošināta konsekvence visā sistēmā. Līdzīgi kā mobilā satura pārvaldības sistēma, arī komponentu satura pārvaldības sistēma var būt veidota atsevišķi vai būt kādas lielākas sistēmas sastāvdaļa. Piemērs – Vasont (http://www.vasont.com/vasont/).

Kā redzams, šis dalījums nav savstarpēji izslēdzošs, jo viena liela uzņēmuma satura pārvaldības sistēma var iekļaut visus minētos veidus, turklāt šo robežu noteikšanu apgrūtina satura pārvaldības ietvari, kas ļauj sistēmu ar nelielu sākotnējo funkcionalitāti paplašināt līdz visaptverošai pārvaldības sistēmai.

2.2. Nozīmīgākās tīmekļa satura pārvaldības sistēmas

No 2.1 apakšnodaļā minētajiem satura pārvaldības sistēmu veidiem, turpmāk tiks apskatītas tikai tīmekļa satura pārvaldības sistēmas, jo citādi datu apmaiņas formāts var kļūt pārāk plašs un nelietojams. Tīmekļa satura pārvaldības sistēmu izvēlei ir gan objektīvi, gan subjektīvi iemesli. Pirmkārt, izvēloties tīmekļa satura pārvaldības sistēmas, ir lielākas izredzes, ka datu apmaiņas formāts tiks izmantota praktiski, jo WCMS ir plašāk izplatītas, nekā cita veida CMS. Otrkārt, WCMS ir vienveidīgāki dati, piemēram, salīdzinot ar ECMS, kas ļauj tos labāk apkopot un izmantot datu apmaiņas formāta izveidē. Kā subjektīvais iemesls, jāmin lielāka autora personiskā pieredze, strādājot ar tīmekļa satura pārvaldības sistēmām.

Šobrīd nozīmīgākās atvērtā koda satura pārvaldības sistēmas ir Drupal, Joomla, un Wordpress, turklāt pārējās CMS ievērojami atpaliek no šī vadošā trijnieka [13]. Visas šīs sistēmas ir paplašināmas, tām eksistē milzīgs daudzums jau gatavu paplašinājumu, turklāt tām visām ir pieejams API, kas ļauj viegli izveidot individuāli pielāgotus risinājumus. Šos iemeslu dēļ gan šajā apakšnodaļā, gan turpmāk tiks apskatīta tikai tā funkcionalitāte, kas ir pieejama uzstādot šo sistēmu standarta instalācijas.

2.2.1. Drupal

Drupal ir bezmaksas atvērtā koda satura pārvaldības sistēma un satura pārvaldības ietvars [14], kas cenšas apvienot divus konfliktējošus mērķus – elastīgumu un vienkāršību. Ja sistēma ir pārāk vienkārša, tā var tikt izmantota tikai vienā veidā, savukārt, ja tā ir pārāk elastīga, tā var būt pārāk sarežģīta, lai to apgūtu jauni lietotāji. Tādēļ Drupal ir kā konstruktors, tikko uzstādītai Drupal sistēmai, minimālajā komplektācijā ir iespējoti tikai daži nepieciešamākie moduļi, tomēr pievienojot vai izstrādājot jaunus moduļus ir iespējams uzbūvēt sistēmu, kas atbilst nepieciešamajām prasībām. Šajā darbā tiek izmantota Drupal 7.0 versija. Drupal tīmekļa vietnē ir pieejami 7975 moduļi sistēmas paplašināšanai, t.s. 1585 moduļi, kas ir savietojami ar 7.0 versiju (2011. gada 8. maija dati).

2.2.2. Joomla

Joomla gluži kā Drupal ir bezmaksas atvērtā koda satura pārvaldības sistēma [15], tomēr Joomla nepiemin vārdus satura pārvaldības ietvars, kaut arī ir paplašināma un pielāgojama, līdzīgi kā Drupal. Šajā darbā tiek izmantota Joomla 1.6.1 versija. Joomla oficiālajā paplašinājumu katalogā ir pieejami 7504 paplašinājumi, no kuriem 1580 ir savietojami ar Joomla 1.6 (2011. gada 8. maija dati).

2.2.3. Wordpress

Wordpress ir viena no populārākajām satura pārvaldības sistēmām, kas no vienkāršas tīmekļa žurnālu sistēmas ir attīstījusies līdz tādai pakāpei, ka to izmanto kā pilnvērtīgu satura pārvaldības sistēmu, kas spēj pārvaldīt ne tikai rakstus un kategorijas, bet visa veida digitālo saturu [16]. Tas lielā mērā ir iespējams, pateicoties bagātīgajam paplašinājumu klāstam (2011. gada 8. maijā - 14305 paplašinājumi). Šajā darbā tiek izmantota Wordpress 3.1.2 versija.

2.3. Biznesa dati

Neitrāls datu apmaiņas formāts nedrīkst iekļaut CMS specifiskus datus, jo citādi šis formāts nemaz nebūs neitrāls. Turklāt iekļaujot CMS specifiskus datus, formāts var uzblīst un kļūt nelietojams, jo ļoti iespējams, ka satura pārvaldības sistēmas nespēs apstrādāt citas satura pārvaldības sistēmas datus, vai to varēs tikai daļēji. Tas noved pie loģiska jautājuma – kas ir CMS specifiski dati, un kuri ir tie dati, kas ir jāiekļauj datu apmaiņas formātā? Kā atbilde nāk prātā jēdziens biznesa dati.

Kas tad īsti ir biznesa dati? Precīzu atbildi autoram atrast neizdevās, bet tuvākais rezultāts ir Google pirmais rezultāts atslēgvārdiem „business data definition" – „Data about people, places, things, business rules, and events used to operate a business. It is not metadata". Diemžēl šī definīcija ir tikai Google saglabātajos rezultātos, jo vietne, kura piedāvā šo definīciju vairs nav pieejama. Rupji tulkojot šo definīciju, par biznesa datiem var saukt visu informāciju, kas tiek izmantota biznesa vajadzībām, tomēr ir pieminēts, ka tie nav metadati. Šajā gadījumā metadati ir dati, kas apraksta biznesa datus, jeb CMS specifiskie dati. Savukārt metadati, kas tiek pievienoti tīmekļa lapai, izmantojot HTML meta tagu, ja vien tas nenotiek pilnīgi automātiski, ir uzskatāmi par biznesa datiem, jo tie tiek izmantoti meklēšanas dzinēju optimizācijā.

Joprojām ir jāatbild uz jautājumu, kā atšķirt datus, kurus nepieciešams migrēt? Autors piedāvā migrēt tikai tos datus, kurus lietotājs ievada satura pārvaldības sistēmā izmantojot CMS piedāvāto saskarni. Tūdaļ rodas jautājums par datiem, kas tiek importēti sistēmā bez manuālas datu ievades. Balstoties uz autora pieredzi var izšķirt divus gadījumus:

1) dati tiek importēti, izmantojot CMS iebūvētos līdzekļus;
2) dati tiek importēti, izmantojot CMS paplašinājumus vai pielāgotas procedūras.

Pirmajā gadījumā tipiski netiek importēti tādi dati, kurus nav iespējams ievadīt manuāli, tātad no datu viedokļa šo jautājumu nav nepieciešams apskatīt, savukārt otrais gadījums, kā tas minēts iepriekš, šajā darbā netiek apskatīts, jo šādā gadījumā tāpat ir nepieciešama individuāla pieeja.

Arī migrējot tikai manuāli ievadītus datus nevar novilkt stingru robežu starp abiem datu veidiem, jo arī CMS specifiskus datus ir iespējams ievadīt manuāli. Kā piemēru var minēt raksta lauku statuss. Vienā CMS šim laukam var būt vairākas vērtības, piemēram, publicēts, skice, nepublicēts, dzēsts, savukārt citā satura pārvaldības sistēmā šim laukam var būt tikai divas vērtības – publicēts, nepublicēts, vai arī šāda lauka var nebūt vispār. Ko darīt? Šis lielā mērā ir subjektīvs jautājums – ja šāds lauks ir sastopams visās apskatītajās CMS, tad to ir vērts iekļaut datu apmaiņas formātā, bet rūpīgi izvēloties lauka vērtības tā, lai tās neradītu konfliktus migrācijas procesā.

Augstāk minēto iemeslu dēļ, datu apmaiņas formātam ir jāparedz iespēja pievienot CMS specifiskus datus, tomēr tie nedrīkst būt iekļauti formāta definīcijā. Šādā veidā eksportējot un importējot datus ir iespējams nodot vairāk informācijas, nekā definēts datu apmaiņas formātā, tomēr tas prasa eksporta un importa procedūru pielāgojumus.

2.4. Modularitāte

Apskatītās satura pārvaldības sistēmas ir modulāras, un sastāv no sistēmas dziņa un dažādiem paplašinājumiem, kurus katrā CMS dēvē savādāk. Drupal tie ir moduļi, Joomla – komponenti, savukārt Wordpress tie tiek dēvēti par spraudņiem. Šie paplašinājumi tiek dalīti vēl sīkāk, piemēram, Joomla satura pārvaldības sistēmā paplašinājumi tiek dalīti komponentos, moduļos un spraudņos, bet vienkāršības dēļ turpmāk šajā darbā visi paplašinājumi tiks dēvēti par komponentiem.

CMS komponentus var sadalīt trīs grupās:

1) komponenti, kas ir neatņemama CMS sastāvdaļa;
2) komponenti, kas ir iekļauti CMS sastāvā, bet kurus var atspējot un aizstāt ar citiem;
3) pārējie komponenti, kas nav iekļauti CMS sastāvā.

Pirmajā grupā ietilpst komponenti, bez kuriem CMS nav spējīga darboties, vai arī sistēmas darbība ir apgrūtināta. Otrajā grupā ir komponenti, kas ietilpst CMS sastāvā, bet neietilpst pirmajā grupā, t.i. bez šiem komponentiem sistēma spēj pilnvērtīgi darboties. Otrās grupas komponenti pēc noklusējuma var būt gan iespējoti, gan atspējoti, un no trešās grupas tos atšķir tikai tas, ka tie ir iekļauti CMS sastāvā, un tādēļ ir „oficiālie” komponenti. Tomēr šī atšķirība ir būtiska, jo neatkarīgie komponentu izstrādātāji var paļauties uz to, ka šie komponenti ir CMS sastāvā, un tādēļ izmantot tos kā bāzi savu komponentu izstrādei, vai arī izmantot šo komponentu tabulas datubāzē, lai uzglabātu datus. Savukārt trešajā grupā ietilpst visi pārējie komponenti, un tos var izmantot gan CMS funkcionalitātes paplašināšanai, gan

esošās funkcionalitātes uzlabošanai. Trešā grupa skaitliski ir vislielākā, jo tajā ietilpst visi trešo pušu izstrādātie paplašinājumi, kuru skaits, kā jau minēts, sniedzas tūkstošos.

Komponentus var dalīt tālāk, atkarībā no tā, vai tie uzglabā datus, vai tikai izmanto tos, piedāvājot savādāku funkcionalitāti. Kā piemēru var minēt rakstu pārvaldes komponentu, kas ļauj pievienot un parādīt jaunu rakstu, pret komponentu, kas parāda desmit jaunākos rakstus. Tāpat komponentus var dalīt pēc tā, vai tie ir CMS specifiski, vai nē, piemēram, CMS spraudņu pārvaldnieks pret kategoriju pārvaldnieku. Ņemot vērā to, ka mērķis ir apkopot datus tīmekļa satura migrācijas valodai, ir jāapskata tikai tie komponenti, kas nav CMS specifiski un uzglabā biznesa datus.

2.5. Tīmekļa satura pārvaldības sistēmu dati

Šīs apakšnodaļas mērķis ir apkopot datus, kurus iespējams manuāli ievadīt Drupal, Joomla un Wordpress satura pārvaldības sistēmās, lai iegūtu informāciju, ko izmantot tīmekļa satura iezīmēšanas valodas veidošanā.

Lai apkopotu datus tika apskatītas datu ievades un rediģēšanas formas, un šajās formās tika atlasīti lauki, kuros ir biznesa dati, un lauki, kuros ir potenciāli noderīgi CMS specifiski dati, kuri ir vai varētu būt sastopami arī citās satura pārvaldības sistēmās. Tāpat tika apskatīti arī pārējā formā esošā informācija, un nepieciešamības gadījumā pievienota pārējiem datiem. Apkopojot datus netika apskatīta CMS specifiska informācija un komponenti, piemēram, CMS konfigurācija, paplašinājumu pārvalde, tēmas, izskata opcijas un lietotāju tiesības.

Aprakstot datu ievades un rediģēšanas formas ļoti viegli var novirzīties no datu apkopošanas, tā vietā apskatot katru lauku, un šādā veidā izveidojot CMS dokumentāciju, kas, savukārt, nav šī darba mērķis. Tādēļ šajā apakšnodaļā nav aprakstīti pilnīgi visi ievades lauki un cita veida informācija par apskatīto formu, bet gan pārskaitīti tikai tie lauki, kas satur vai nu biznesa, vai cita veida informāciju, ko, iespējams, ir vērts iekļaut migrējamajos datos.

Veicot šo darbu ir vērts izpētīt datu glabāšanas veidu datubāzē, jo šādā veidā ir iespējams iegūt informāciju par datu modeli un struktūru satura pārvaldības sistēmā, kā arī citu, potenciāli noderīgu, informāciju. Šajā apakšnodaļā, lai norādītu informācijas atrašanās vietu datubāzē tiek izmantota šāda notācija:

tabulas_nosaukums.kolonnas_nosaukums[where_klauzula]#šūnas_atslēga$^{piezīmes\_nr}$

- **tabulas_nosaukums** ir CMS datubāzes tabulas nosaukums, iekļaujot noklusēto prefiksu
- **kolonnas_nosaukums** ir dotās tabulas kolonnas nosaukums

- **where_klauzula** ir SQL valodas WHERE klauzulai līdzīga konstrukcija, ko var izmantot rindu filtrēšanai
- **šūnas_atslēga** ir identifikators, kas nepieciešams vērtības noteikšanai, ja šūnas dati ir saglabāta struktūra, piemēram, JSON vai PHP serializēts objekts;
- **piezīmes_nr** – piezīmes numurs ir pievienots, ja par lauku ir norādīta sīkāka informācija tālāk tekstā.

Notācija ir paredzēta, lai palīdzētu atrast nepieciešamos informāciju datubāzē, tādēļ tā var nebūt absolūti precīza norāde uz vajadzīgajiem datiem, it sevišķi, ja dati ir sadalīti vairākās tabulās, kā daudz-pret-daudz gadījumā.

2.5.1. Raksti

Rakstu pārvalde ir praktiski jebkuras CMS neatņemama sastāvdaļa, un apskatītās sistēmas nav izņēmums.

Wordpress ir divu veidu raksti – ziņojumi (*posts*) un lappuses (*page*s), kuri no datu viedokļa savā starpā atšķiras tikai ar to, ka lappusēm nav lauka izvilkums (*excerpt*), un lappusēm nevar norādīt kategorijas un tagus. Līdzīga situācija ir Drupal satura pārvaldības sistēmā, arī tajā eksistē divu veidu raksti – raksti (*articles*) un vienkāršas lapas (*basic pages*), un arī šajā gadījumā atšķirības starp abiem veidiem ir ļoti līdzīgas – lapām nav iespējams pievienot ievada attēlu, un nav iespējams pievienot tagus. Savukārt Joomla satura pārvaldības sistēmā raksti nav dalīti sīkāk.

Gan Wordpress, gan Drupal sistēmās ir iebūvēta rakstu versiju kontroles sistēma, bet Joomla ir iespējams to pievienot, izmantojot trešās puses paplašinājumus.

Joomla un Drupal atbalsta satura ievadi vairākās valodās, savukārt Wordpress nav iebūvēts vairāku valodu atbalsts.

No apskatītajām sistēmām, tikai Joomla piedāvā rakstiem ievadīt HTML metadatus, bet visās sistēmās, ieskaitot Joomla, eksistē paplašinājumi, kas šos metadatus var ģenerēt automātiski no raksta informācijas.

Visās apskatītajās sistēmās ir iespējams rakstu sadalīt ievadā un pamattekstā, tikai atšķiras ievades un datu glabāšanas veids. Wordpress un Joomla ievadei izmanto pogu teksta redaktorā, kas kursora atrašanās vietā ievieto atzīmi, ka beidzas ievads, un sākas pamatteksts. Drupal savukārt tie ir divi atsevišķi teksta lauki. Šo datu glabāšanas veids ir atšķirīgs katrā sistēmā – Wordpress gan ievads, gan pamatteksts tiek glabāti vienā laukā, bet Drupal un Joomla divos laukos. Ja ievads nav norādīts, tad Drupal visu raksta saturu glabā pamattekstam paredzētajā laukā, savukārt Joomla visu saturu glabā ievadam paredzētajā laukā.

Sīkāks rakstu datu salīdzinājums ir dots 2.1 tabulā.

2.1. tabula

Rakstu datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	**Wordpress**	**Drupal**	**Joomla**
Virsraksts	`wp_posts.post_title`	`node.title`	`jos_content.title`
Ievads	`wp_posts.post_content`[1]	`field_data_body.body_summary`	`jos_content.introtext`
Pamatteksts	`wp_posts.post_content`[1]	`field_data_body.body_value`	`jos_content.fulltext`[2]
Izvilkums	`wp_posts.post_excerpt`	-	-
Autors	`wp_posts.post_author`	`node.uid`	`jos_content.created_by`
Izveidots	`wp_posts.post_date_gmt`	`node.created`	`jos_content.created`
Aizstājvārds	`wp_posts.post_name`	`url_alias.alias (source = 'node/' + id)`[3]	`jos_content.alias`
Kategorijas	`wp_term_relationships.term_taxonomy_id`	-	`jos_content.catid`
Tagi	`wp_term_relationships.term_taxonomy_id`	`taxonomy_index.tid`	-
Statuss	`wp_posts.post_status`	`node.status`	`jos_content.state`
Rādīt pirmajā lapā	`wp_options.sticky_posts`[2]	`node.promote`	`jos_content.featured`
Valoda	-	`node.language`	`jos_content.language`
Autora aizstājvārds	-	-	`jos_content.created_by_alias`
Publicēt no	-	-	`jos_content.publish_up`
Publicēt līdz	-	-	`jos_content.publish_down`
Atsauces notifikācija (*trackback*)	`wp_posts.to_ping`	-	-
Revīzijas žurnāla piezīme	-	`node_revision.log`	-
Atļaut komentārus	`wp_posts.comment_status`	`node.comment`	-

2.1. tabulas komentāri:

1) `wp_posts.post_content` lauks satur gan ievadu, gan pamattekstu, kas ir atdalīts ar tekstu `<!--more-->`
2) ja nav norādīts raksta ievads, tad saturs tiek glabāts `jos_content.introtext` laukā;
3) aizstājvārdi Drupal tiek glabāti `url_alias` tabulā, piemēram, raksta ar `id` vērtību 1 aizstājvārds atrodas `alias` kolonnā, kur `source` kolonnas vērtība ir „node/1”.

2.5.2. Taksonomija

Viena no satura pārvaldības pamatfunkcijām ir satura strukturēšana. Visas apskatītās CMS piedāvā satura strukturēšanu, un šajās sistēmās ir iespējams tikai hierarhisks vai lineārs struktūru modelis, t.i. netiek pieļauta situācija, kad elementa bērni satur doto elementu kā bērnu.

Wordpress struktūra tiek veidota izmantojot kategorijas un tagus, Drupal tiek izmantoti tagi, bet Joomla – kategorijas. Wordpress gadījumā tagi un kategorijas atšķiras ar to, ka kategorijām var būt vecāks, t.i. kategorijas ir hierarhiskas, bet tagi nav. Gan Drupal tagi, gan Joomla kategorijas ir hierarhiskas struktūras. Wordpress un Drupal atļauj vienam satura ierakstam piešķirt vairākas kategorijas un tagus, bet Joomla ļauj izvēlēties tikai vienu kategoriju. Jāatgādina gan, ka tiek apskatītas tikai standarta instalācijas, jo šo Joomla trūkumu var labot ar paplašinājumu palīdzību.

Ir redzams, ka nosaukumi tags un kategorija tiek lietoti juceklīgi, un no iepriekšējās rindkopas nav iespējams saprast, kas ir kas. Wordpress atbalsta dokumentācija [17] palīdz saprast atšķirību starp kategorijām un tagiem – kategorijas ir galvenais satura grupēšanas elements, savukārt tagi ir paredzēti, lai aprakstītu saturu detalizētāk. Piemēram, raksts par satura migrāciju starp CMS var būt kategorijās „Satura pārvaldības sistēmas" un „Datu migrācija", bet, lai sīkāk aprakstītu raksta saturu var izmantot tagus PHP, XML un XPath, kas nosauc izmantotās tehnoloģijas. Ir redzams, ka pēc šī iedalījuma Drupal tagi patiesībā ir kategorijas, jo tie ir vienīgais satura strukturēšanas veids, un turklāt tie ir hierarhiski.

Apskatīto CMS kategoriju un tagu datu salīdzinājums ir dots tabulās 2.2 un 2.3. Jāatzīmē, ka ir iespējamas vairāku veidu kategorijas, kas atkarīgas no satura tipa. Piemēram, rakstiem ir definēts viens kategoriju koks, savukārt saišu sarakstam – cits. Kategoriju veidus, jeb taksonomiju Wordpress nosaka pēc `wp_term_taxonomy.taxonomy` lauka – ja tas ir „category", tad tā ir rakstu kategorija, ja „post_tag", tad rakstu tags, ja „link_category", tad saišu kategorija, utt. Drupal šādu funkcionalitāti atļauj `taxonomy_vocabulary` tabula, savukārt Joomla šo informāciju glabā `jos_categories.extension` laukā.

2.2. tabula

Kategoriju datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	**Wordpress**	**Drupal**	**Joomla**
Nosaukums	`wp_terms.name`	-	`jos_categories.title`
Aizstājvārds	`wp_terms.slug`	-	`jos_categories.alias`
Vecāks	`wp_term_taxonomy.parent`	-	`jos_categories.parent_id`
Apraksts	`wp_term_taxonomy.description`	-	`jos_categories.description`

2.3. tabula

Tagu datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	Wordpress	Drupal	Joomla
Nosaukums	wp_terms.name	taxonomy_term_data.name	-
Aizstājvārds	wp_terms.slug	url_alias.alias	-
Vecāks	-	taxonomy_term_hierarchy.parent	-
Apraksts	wp_term_taxonomy.description	taxonomy_term_data.description	-

2.5.3. Izvēlnes

Kategorijas un tagi nav vienīgais veids, kā strukturēt informāciju tīmekļa vietnē. Mazas vietnes var iztikt arī bez kategoriju un tagu lietošanas, toties praktiski nav sastopamas vietnes bez navigācijas izvēlnēm. Izvēlnes pēc būtības ir hierarhiska saišu kopa. Saites var sadalīt divās daļās – iekšējās un ārējās. Ārējās saites norāda uz resursiem, kas atrodas ārpus vietnes, bet iekšējās saites norāda uz dažādiem vietnes komponentiem, piemēram, rakstiem, kategorijām vai pat citiem izvēlnes elementiem.

Visās apskatītajās sistēmās izvēlņu pārvaldība ir ļoti līdzīga – ir iespējams izveidot vairākas izvēlnes, un pēc tam izvēlnēm pievienot ierakstus. Informācija par uzglabātajiem izvēlņu datiem ir parādīta tabulā 2.4, savukārt informācija par izvēlņu ierakstu datiem ir tabulā 2.5. Wordpress satura pārvaldības sistēmā izvēlnes tiek glabāta līdzīgi, kā kategorijas, t.i. `wp_term_taxonomy.taxonomy` laukā ir norādīts taksonomijas tips „nav_menu”, izvēlnes tiek glabātas tieši tāpat kā kategorijas, savukārt izvēlņu ieraksti tiek glabāti vienā tabulā ar rakstiem.

2.4. tabula

Izvēlņu datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	Wordpress	Drupal	Joomla
Nosaukums	wp_terms.name	menu_custom.title	jos_menu_types.title
Aizstājvārds	wp_terms.slug	menu_custom.menu_name	jos_menu_types.menutype
Apraksts	-	menu_custom.description	jos_menu_types.description

2.5. tabula

Izvēlņu ierakstu datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	Wordpress	Drupal	Joomla
Ieraksta nosaukums	wp_posts.post_title	menu_links.link_title	jos_menu.title

Ieraksta apraksts	wp_posts. post_excerpt	menu_links. options[1]	jos_menu.params# menu- anchor_title
Ieraksta ceļš	wp_postmeta. meta_value[meta_key= _menu_item_url]	menu_links. link_path	jos_menu.link
Vecāks	wp_postmeta. meta_value[meta_key= _menu_item_ menu_item_parent]	menu_links. (p1-p9)[2]	jos_menu. parent_id
Secība	wp_posts. menu_order	menu_links.weight	jos_menu. ordering
Statuss	-	menu_links.hidden	jos_menu. published
Izvēlne	wp_term_relationships. term_taxonomy_id	menu_links. menu_name	jos_menu. menutype

2.5. tabulas komentāri:

1) `options` kolonna satur PHP serializētu objektu, kurš turklāt tiek glabāts binārā formā;
2) Drupal izvēlnes ieraksta vecāks tiek norādīts pa līmeņiem, pirmā līmeņa vecāka ārējā atslēga ir kolonnā p1, otrā – p2, utt., līdz p9, tādēļ Drupal atbalsta izvēlnes ar dziļumu līdz 9, kas praktiskiem mērķiem ir vairāk nekā pietiekoši; savukārt paša izvēlnes ieraksta dziļums tiek norādīts `menu_links.depth` laukā.

2.5.4. Lietotāji

Katrā satura pārvaldības sistēmā ir vismaz viens lietotājs – administrators, kuram ir pilnas sistēmas lietošanas tiesības. Tomēr tipiski CMS piedāvā lietotāju pārvaldību, un ļauj izveidot vairāk nekā vienu lietotāju, kuram kā minimums ir unikāls lietotājvārds un parole, un piešķirtas kādas noteiktas tiesības. Lai būtu vieglāk pārvaldīt tiesības, CMS izmanto grupas, kurām piešķir noteiktas tiesības. Pēc tam pievienojot lietotāju kādai grupai lietotājs saņem visas grupas tiesības.

No migrācijas viedokļa lietotāji un to tiesības nav vienkārša datu grupa. Pirmkārt, vispārīgā gadījumā nav iespējams migrēt lietotāju paroles, ja vien tās netiek glabātas atklātā tekstā, kas savukārt ir slikta prakse. Protams, ja paveicas, un gan avota, gan mērķa sistēmas izmanto vienādu paroļu glabāšanas tehniku, tad tā nav problēma, bet tas nav vispārīgs gadījums. Otrkārt, ir problemātiski pārnest lietotāju grupas, jo tās var būt iepriekš definētas, un nerediģējamas, kā tas ir Wordpress gadījumā, un treškārt, tiesības ir specifiskas satura pārvaldības sistēmām.

Ņemot vērā augstāk minēto, autors uzskata, ka migrācijas procesā ir vērts pārnest tikai lietotāju datus (skat. tabulu 2.6). Savukārt lietotāju paroļu jautājumu var atrisināt uzģenerējot jaunu paroli, un aizsūtot to uz e-pastu, vai kā citādi to paziņojot lietotājam.

2.6. tabula

Lietotāju datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	Wordpress	Drupal	Joomla
Lietotājvārds	wp_users. user_login	users.name	jos_users. username
Vārds	wp_usermeta. meta_value [meta_key=first_name]	-	jos_users.name
Uzvārds	wp_usermeta. meta_value [meta_key=last_name]	-	jos_users.name
Iesauka	wp_usermeta. meta_value [meta_key=nickname]	-	-
E-pasts	wp_users. user_email	users.mail	jos_users.email
Tīmekļa vietne	wp_users.user_url	-	-
Informācija par lietotāju	wp_usermeta. meta_value [meta_key=description]	-	-
Reģistrācijas datums	wp_users. user_registered	users.created	jos_users. registerDate
Statuss	wp_users. user_status	users.status	jos_users.block
Laika zona	-	users.timezone	jos_users.params

2.5.5. Komentāri

Lai tīmekļa vietnes apmeklētāji varētu izteikt viedokli par vietnē ievietoto saturu, daudzas CMS paredz satura komentēšanas iespēju. Wordpress un Drupal šāda iespēja ir jau standarta instalācijā, savukārt Joomla satura pārvaldības sistēmā ir nepieciešams instalēt papildus paplašinājumu. Wordpress un Drupal arhitektūra ir veidota tā, ka ir iespējams komentēt jebkuru ievietoto saturu. 2.7. tabulā ir attēlots komentāru datu salīdzinājums starp apskatītajām sistēmām.

2.7. tabula

Komentāru datu salīdzinājums izvēlētās satura pārvaldības sistēmās

Nosaukums	Wordpress	Drupal	Joomla
Vārds	wp_comments. comment_author	-	-
Lietotājs	wp_comments.user_id	comment.uid	-
E-pasts	wp_comments. comment_author_email	comment.mail	-

Mājas lapa	wp_comments. comment_author_url	comment.homepage	-
Temats	-	comment.subject	-
Komentārs	wp_comments. comment_comment	field_data_comment_body. comment_body_value	-
Komentāra laiks	wp_comments. comment_date_gmt	comment.created	-
Statuss	wp_comments. comment_approved	comment.status	-
Atbilde uz	wp_comments. comment_parent	comment.pid	-
Komentāra objekts	wp_comments. comment_post_id	comment.nid	-

2.5.6. Citi dati

Tīmekļa satura pārvaldības sistēmās, gan noklusētajā instalācijā, gan ar paplašinājumu palīdzību, ir pieejams arī cita veida saturs un tā pārvaldība, kā piemērus var minēt reklāmkarogu, saišu un multivides failu pārvaldību. Tomēr šis saturs nav tik izplatīts, salīdzinot ar iepriekš minētajiem datiem. Piemēram, reklāmkarogu pārvaldību pēc noklusējuma piedāvā tikai Joomla, bet multivides failu pārvaldību – tikai Wordpress (Joomla ir multivides pārvaldnieks, bet tas pilda tikai failu pārlūka funkcijas). Šī iemesla dēļ šajā darbā citi satura pārvaldības sistēmu dati netiek apskatīti.

3. IEZĪMĒŠANAS VALODAS

Iezīmēšanas valoda ir standarta teksta kodēšanas sistēma, kas sastāv no simbolu kopas, kas ir ievietota tekstā, lai kontrolētu tā struktūru, formatējumu vai attiecības starp tā daļām [18]. Šīs nodaļas pirmajā pusē ir īsi pastāstīta iezīmēšanas valodu attīstības vēsture, apskatīta iezīmēšanas valodu taksonomija un pastāstīts par valodas interpretatoru nozīmi. Nodaļas turpinājumā sīkāk tiek apskatīts, kā iespējams paplašināt XML (*eXtensible Markup Language*), lai izveidotu jaunu iezīmēšanas valodu, un parādītas XML validācijas iespējas.

3.1. Īsa vēsture

Termins iezīmēšana (*markup*) radies no izdevniecību prakses „iezīmēt" manuskriptus, pievienojot instrukcijas, kā tiem būtu jāizskatās drukātā veidā. Datoru pasaulē priekšstatu par iezīmēšanas valodu attīstību ir iespējams gūt apskatot teksta redaktoru attīstību.

No sākuma bija iespējams rediģēt tikai „tīru" tekstu, t.i. nebija pieejama teksta formatēšana, un redaktors pēc būtības līdzinājās rakstāmmašīnai. Pēc tam parādījās vienkāršas teksta formatēšanas iespējas – treknraksts, slīpraksts, pasvītrojums un cits formatējums, ko spēja attēlot tā laika printeri, un teksta faili kļuva par dokumentiem. Lai datorā attēlotu šīs iespējas, teksta redaktoru izstrādātāji izgudroja speciālus kodus, jeb tagus, kas bija jāieliek pirms un pēc vārdiem, kurus bija nepieciešams formatēt, piemēram, iezīmēt vārdu treknrakstā varēja pirms un pēc tā uzrakstot „^B" (bez pēdiņām). Pirmais ^B ieslēdza treknrakstu, bet otrais – izslēdza. Attīstoties datoriem, kļuva iespējams formatējumu attēlot tieši uz ekrāna, bez kodu palīdzības, un radās pirmie WYSIWYG (*what you see is what you get*) redaktori. Tas gan nenozīmē, ka šie kodi pazuda, tie vienkārši netika attēloti uz ekrāna, un tādēļ zuda nepieciešamība tos attēlot cilvēkam saprotamā formā, kas noveda pie binārajām iezīmēšanas valodām.

Redaktori kļuva arvien attīstītāki, un piedāvāja arvien vairāk iespēju, tomēr organizācijām ar lieliem katalogiem un dokumentu vadības sistēmām, dokumentu izskats bija otršķirīgs, tām vairāk interesēja tas, lai dokumentiem būtu struktūra, ko spētu nolasīt un izmantot citas programmas. Ideja, ka iezīmēšanas valodām ir jāattēlo dokumenta struktūra, bet vizuālais attēlojums ir jāatstāj valodas interpretatora ziņā noveda pie SGML (*Standard Generalized Markup Language*) radīšanas. SGML specificēja sintaksi gan iezīmējuma iekļaušanai dokumentos, gan atsevišķu sintaksi (DTD – *Document Type Definition*), lai aprakstītu, tagus un to atrašanās vietu dokumentā. Tas ļāva izstrādātājiem izveidot un lietot

tādus tagus un struktūru, kādu tie vēlējās. Tādēļ precīzāk būtu SGML saukt par metavalodu, no kuras ir atvasinātas citas iezīmēšanas valodas.

Viena no valodām, kas ir atvasināta no SGML ir HTML (*HyperText Markup Language*). HTML, pateicoties tā elastībai un vienkāršībai, bija viens no tīmekļa panākumu stūrakmeņiem. Tomēr, par spīti tā panākumiem, HTML tomēr ir pārāk ierobežota valoda, jo tā ir piemērota teksta attēlošanai pārlūka, bet ne plašākiem pielietojumiem. Tādēļ tika izveidota XML valoda, kuras mērķis bija vienkāršot SGML, fokusējoties uz dokumentiem internetā. XML ir atvasināta no SGML, un tā joprojām ir metavaloda, jo ļauj izstrādātājiem veidot savu valodu uz tās bāzes. 2000. gadā kā pamatu ņemot HTML, tika izveidota XHTML (*eXtensible HyperText Markup Language*) valoda, kas ir atvasināta nevis no SGML, bet gan no XML, un tādēļ ir vienkāršāk apstrādājama, gan no pārlūku, gan citu lietojumprogrammu puses.

3.2. Interpretatoru nozīme

Teksta iezīmējums ir valoda, un pats no sevis tas neko nedara. Gluži kā manuskriptos iezīmētais teksts pats no sevis neiegūst formatējumu drukātā formātā, ir jābūt kādam, kas šo iezīmējumu saprot, un attiecīgi apstrādā iezīmēto tekstu. Iezīmēšanas valodu gadījumā to dara interpretators. Šis ir ļoti svarīgs aspekts, jo vienai valodai var būt daudzi interpretatori. Turklāt katrs no šiem interpretatoriem var apstrādāt doto valodu mazliet savādāk.

Ja salīdzina HTML un XML, tad HTML ir definēta valoda, kurai eksistē vairāki interpretatori (pārlūki), kas attēlo iezīmēto saturu. Savukārt XML ir metavaloda. Ir pieejami rīki XML lasīšanai un rakstīšanai, bet interpretatora izveide ir izstrādātāja ziņā, jo tikai izstrādātājs zina pareizo interpretāciju katram izveidotajam tagam.

3.3. Iezīmēšanas valodu taksonomija

Visas iezīmēšanas valodas pievieno tekstam kodus, lai iezīmētu tā formātu un struktūru, tomēr katra iezīmēšanas valoda šo uzdevumu veic mazliet savādāk. Iezīmēšanas valodas atšķiras dažādos aspektos [19]:

- tagu nozīme – vienām valodām tagi nosaka formatējumu, citām – apraksta struktūru, vēl citām tie kalpo kā komandas;
- tagu kodējums - dažas valodas kā tagus izmanto vienkāršu tekstu, bet citas – bināros kodus;
- valodas paplašināmība - dažām valodām ir stingri fiksēta tagu kopa, savukārt citas ļauj veidot savus tagus;

- pārklājuma diapazons - dažām valodām ir ļoti ierobežots tagu skaits, citām tas ir lielāks.

3.3.1. Tagu nozīme

Iezīmēšanas valodas vienlaicīgi var saturēt dažādus tagus, kas pieder dažādiem iezīmējuma veidiem. Pēc tagu nozīmes var izdalīt trīs veidu iezīmējumus [20].

Attēlošanas iezīmējums tiešā veidā nosaka teksta formatējumu, piemēram, HTML `<b>` tagam ir tieši šāda nozīme. Tas skaidri pasaka – visu tekstu, kas atrodas starp atverošo `<b>` un aizverošo `</b>` tagu, ir jāattēlo treknrakstā. Šo iezīmējuma veidu izmanto arī WYSIWYG teksta redaktori.

Procedurālais iezīmējums ir tekstā ievietotas komandas valodas interpretatoram. Populāras procedurālā iezīmējuma valodas ir LaTeX un PostScript. Šo valodu interpretators parsē tekstu, un izpilda sastaptās instrukcijas. Populāras procedurālā iezīmējuma sistēmas iekļauj programmēšanas konstrukcijas, kas ļauj nodefinēt un izpildīt makro komandas. Piemēram, var nodefinēt instrukciju virkni, kas norenderē 16 punktus lielu treknrakstu un nosaukt to „virsraksts”. Tas noved pie nākošā iezīmējuma veida...

Aprakstošais iezīmējums ļauj pateikt, kas ir dotais teksta gabals, nevis tikai, ko ar to darīt. Šādā veidā ar tagiem iezīmēt dokumenta struktūru, un, nepieciešamības gadījumā, tas ļauj viegli mainīt attēlojuma veidu. Piemēram, HTML kodā `<h1>` tags pieder aprakstošajam iezīmējumam, un iezīmē pirmā līmeņa virsrakstu, savukārt `<i>` tags pieder attēlošanas iezīmējumam, jo tas neko nepasaka par tekstu, tikai norāda, ka tas ir jāattēlo slīprakstā.

3.3.2. Tagu kodējums

Pēc tagu kodējuma iezīmēšanas valodas var dalīt:

1) valodās, kas tagu kodē teksta veidā, piemēram ASCII vai unikodā;
2) valodās, kas tagus kodē binārā formā.

Pirmā veida valodas ir cilvēkiem saprotamā formā, un tās ir nolasāmas bez speciāliem dekodētājiem, kas padara šīs valodas universālākas. Otrā veida valodas tagus kodē binārā veidā, tādejādi padarot šos dokumentus mazākus, kas ļauj tos ātrāk apstrādāt un attēlot. Tomēr šī pieeja prasa ciešu sadarbību starp valodas interpretatoru un sintaksi, kas piesaista valodu vienam interpretatoram.

3.3.3. Valodas paplašināmība

Paplašināmas valodas ļauj veidot jaunus tagus dažādiem nolūkiem. Šajā ziņā var viegli atšķirt, piemēram, HTML no XML.

HTML gadījumā ir noteikts skaits, iepriekš definētu, tagu, ar kuru palīdzību ir jāpaveic nepieciešamais uzdevums. Globālā tīmekļa konsorcijs (W3C – *World Wide Web Consortium*) nosaka HTML standartu, un ar katru versiju tam pievieno un izņem no tā tagus, cenšoties aptvert vairākumam nepieciešamo funkcionalitāti. Ja nepieciešamo funkcionalitāti nav iespējams panākt ar speciāli tam paredzētiem tagiem, var nākties izmantot tagus mērķiem, kuriem tie nav paredzēti. Kā lielisks piemērs kalpo `<table>` tagu izmantošana mājas lapu dizaina veidošanai.

Savukārt XML gadījumā nav iepriekš definētu tagu, tā vietā tiek dota iespēja izveidot jebkādus nepieciešamos tagus. Tādā veidā var izveidot pielāgotu tagu komplektu, kas ideāli apraksta nepieciešamo saturu, turklāt ar katru nākošo versiju var pievienot papildus tagus, un nav nepieciešams gaidīt nākamo standarta versiju. Bet arī paplašināmām valodām ir mīnusi, pirmkārt, ir ļoti viegli izveidot pārlieku sarežģītu un nesaprotamu tagu komplektu, ko būs grūti izmantot, un, otrkārt, jo vairāk cilvēku, un sistēmu strādā ar jauno valodu, jo rūpīgāk ir jāplāno katra izmaiņa, kas galu galā noved pie nepaplašināmas iezīmēšanas valodas ar visām tās problēmām.

3.3.4. Pārklājuma diapazons

Iezīmēšanas valodām ir noteikts formatējuma un strukturālo elementu diapazons, ko tās var attēlot. Ja šis diapazons ir zems, valoda atbalsta tikai daļu no potenciāli iespējamajiem tagiem, toties nelielā tagu skaita dēļ, to ir vieglāk iemācīties un realizēt. Ja tagu skaits ir liels, valodas iespējas kļūst plašākas, bet tās kļūst sarežģītāk realizēt.

Atsevišķi ir jāapskata paplašināmas iezīmēšanas valodas, jo tās var saturēt neierobežotu daudzumu nepieciešamo tagu, tādēļ arī to pārklājums ir neierobežots, toties jāņem vērā, ka katram pievienotajam tagam ir nepieciešams realizēt atbalstu interpretatorā.

3.4. XML paplašināšana

XML ir viena no vispopulārākajām iezīmēšanas valodām, tādēļ internetā ir daudz pamācību par to, kā izmantot XML [21], kā arī ir brīvi pieejama XML specifikācija [22]. Tomēr šajā apakšnodaļā ir pieminētas būtiskākās iezīmes, kurām ir nozīme veidojot no XML atvasinātu valodu.

3.4.1. *Labi formēts (well-formed) XML*

XML ļauj izstrādātājiem veidot savas, no XML atvasinātas valodas, tomēr gan valodai, gan tajā veidotajiem dokumentiem jābūt labi formētiem, t.i. jāatbilst XML specifikācijā noteiktajiem sintakses likumiem. To saraksts ir pietiekoši garš, bet daži būtiskākie likumi ir apkopoti tabulā 3.1.

3.1. tabula

Izvēlēti būtiskākie XML sintakses likumi, un to piemēri

XML sintakses likums	Korekts piemērs	Nekorekts piemērs
Ir jābūt vismaz vienam elementam	**<title>** Virsraksts **</title>**	„Virsraksts”

Jābūt unikālam saknes elementam, kas satur visu dokumenta saturu	**<articles>**
	  <title>
	  Virsraksts 1
	  </title>
	  <title>
	  Virsraksts 2
	  </title>
	</articles>
		<title>
		  Virsraksts 1
		</title>
		<title>
		  Virsraksts 2
		</title>

Tagiem jābūt korekti iestarpinātiem – tie nedrīkst pārklāties	<articles>
	  <title>
	  Virsraksts 1
	  </title>
	  <content>
	  Saturs
	  </content>
	</articles>
		<articles>
		  <title>
		  Virsraksts 1
		  <content>
		  </title>
		  Saturs
		  </content>
		</articles>

Tagi ir reģistrjūtīgi	<**title**>Virsraksts</**title**>	<Title>Virsraksts</TITLE>

Atribūtu vērtībām ir jābūt pēdiņās	<title **id=”1”**>
	  Virsraksts
	</title>
		<title id=1>
		  Virsraksts
		</title>

3.4.2. *XML vārdtelpas (namespaces)*

XML vārdtelpas [23] ļauj pievienot jaunus tagus jau esošai XML bāzētai iezīmēšanas valodai, nebaidoties par to, ka šāds tags jau eksistē vai tiks pievienots nākošajās valodas versijās. Piemēram, esošai iezīmēšanas valodai ir nepieciešams pievienot jaunu elementu `raksts`, kas saturēs informāciju pa žurnāla rakstu, bet šajā valodā jau eksistē elements `raksts`, kas satur informāciju par tautiskajiem ornamentiem. Izmantojot vārdtelpas ir iespējams iekļaut dokumentā abus elementus, turklāt nesapludinot to nozīmes. Šī piemēra paraugs ir dots zemāk, un tas satur divas vārdtelpas - `http://andis.komarovs/zurnali` un `http://andis.komarovs/ornamenti`.

```
<root
xmlns:zurnals="http://andis.komarovs/zurnali"
xmlns:ornaments="http://andis.komarovs/ornamenti">
<zurnals:raksts>
```

```
    <zurnals:virsraksts>
        Žurnāla raksta virsraksts
    </zurnals:virsraksts>
</zurnals:raksts>
<ornaments:raksts>
    <ornaments:novads>
        Latgale
    </ornaments:novads>
</ornaments:raksts>
</root>
```

3.4.3. XML validācija

XML validācija ir būtiska iezīmēšanas valodas sastāvdaļa, jo validējot dokumentu ir iespējams pārliecināties par tā atbilstību iezīmēšanas valodas definīcijai, savukārt definīciju var nodefinēt ar shēmu, jeb gramatikas palīdzību. Populārākās validācijas shēmas ir DTD (*Document Type Definition*), XML Schema un RELAX NG (*REgular LAnguage for XML Next Generation*).

DTD ir visvecākā shēmas valoda no minētajām, un tā ir mantota no SGML, bet XML vajadzībām DTD ir novecojusi valoda, jo tā neatbalsta jaunākās XML iespējas, piemēram, vārdtelpas. Cita plaši lietota XML shēmu valoda ir „XML Schema", kā šīs valodas pluss ir jāmin tas, ka tā ir W3C rekomendācija, tomēr tā ir kritizēta par to, ka tā ir pārāk sarežģīta, jo iekļauj ne tikai XML struktūras validāciju, bet validē arī teksta elementus, pārbauda XML integritāti starp atslēgām un atsaucēm uz tām, un veic citus, ar validāciju nesaistītus darbus [24]. Savukārt RELAX NG valoda ir vienkāršāka nekā XML Schema, un tā ir paredzēta tieši XML validācijai. Gan XML Schema, gan RELAX NG ir iespēja shēmu veidot XML formā, bet RELAX NG ir paredzēta arī kompaktāka pieraksta forma.

4. SATURA MIGRĀCIJAS IEZĪMĒŠANAS VALODA

Šajā nodaļā ir aprakstīta tīmekļa satura iezīmēšanas valodas WCML (*Web Content Markup Language*) skice. Valodas mērķis ir atvieglot tīmekļa satura pārvaldības sistēmu datu migrāciju, tādēļ WCML piedāvā vispārēju datu struktūru, kuras veidošanā ir izmantoti 2. nodaļā minētie apsvērumi, un 2.5. apakšnodaļā apkopotie dati. Lai piedāvātās struktūras dēļ valoda netiktu ierobežota, ir paredzētas noteiktas paplašināšanas iespējas, tādēļ WCML var saukt par paplašināmu valodu. Tāpat tiek mēģināts risināt divas tipiskas satura migrācijas problēmas – saišu tulkošanu un nepieejama satura noteikšanu, kas tiek darīts ieviešot iekšēju atsauču sistēmu.

WCML ir atvasināta no XML, tādēļ visiem WCML dokumentiem ir jāatbilst XML 1.0 specifikācijai [22]. WCML saknes elements ir `<wcml>`, tas satur visu dokumenta saturu, un tam ir neobligāts atribūts `version`, kas nosaka valodas versiju. WCML šobrīd ir piešķirts versijas numurs „0.5”. Tas ir tādēļ, ka tīmekļa satura iezīmēšanas valodu vēl nevar uzskatīt par pilnībā pabeigtu, jo autors uzskata, ka būtu jāapskata vēl plašāka WCMS funkcionalitāte, iekļaujot arī populārākos CMS paplašinājumus, kā arī jāveic valodas testēšana, veicot satura migrāciju starp vairākām CMS, t.s. arī starp tādām, kuras nav apskatītas šajā darbā.

1. pielikumā ir dots WCML dokumenta paraugs ar visiem elementiem un atribūtiem.

4.1. Vietne

`<wcml>` saknes elements var saturēt vienu vai vairākus `<site>` elementus, un neobligātu `<files>` elementu, kas sīkāk aprakstīts 4.6. apakšnodaļā.

`<site>` elements ir paredzēts vienas tīmekļa vietnes un vienas valodas saturam. Tas nozīmē, ka, piemēram, tīmekļa vietnes latviešu valodas versija būs iekļauta vienā `<site>` elementā, bet krievu valodas versija citā. Šāda pieeja izvēlēta, jo praksē vienas vietnes versija katrā valodā satur atšķirīgu saturu, un šīs versijas var tikt uztvertas kā vairākas vietnes ar vienādu dizainu. Tā kā var norādīt vairākus `<site>` elementus, nekas neliedz izmantot šo iespēju, lai aprakstītu saturu no vairākām vietnēm vienlaicīgi.

`<site>` var saturēt neobligātu `xml:lang` atribūtu no `xml` vārdtelpas, kas norāda vietnes valodu. Tāpat `<site>` var saturēt neobligātu `<base-url>` elementu, kurā norāda vecās vietnes bāzes adresi, ko importētājs nepieciešamības gadījumā var izmantot, piemēram, lai relatīvo adresi pārvērstu absolūtajā un spētu izmantot saturu no vecās vietnes. Pārējie `<site>` apakšelementi ir aprakstīti citās apakšnodaļās.

4.2. Vietnes struktūra

<taxonomy> ir neobligāts <site> apakšelements, un tas ir paredzēts ierakstu taksonomijai, t.i. visa veida kategorijām, tagiem un cita veida klasifikatoriem. Tas var saturēt divus neobligātus elementus - <categories> un <tags>, kā arī vairākus <terms> elementus.

<categories> elements satur hierarhiskas rakstu kategorijas, <tags> satur lineārus rakstu tagus, savukārt <terms> elementi satur pārējos taksonus, tādēļ tiem ir obligāts name atribūts, kurš norāda taksona nosaukumu. Taksonu nosaukumi nav definēti, jo <terms> ir paredzēts kā vietturis (*placeholder*) CMS definētiem taksoniem. Lai korekti ieimportētu <terms> saturu var būt nepieciešama pielāgota importa procedūra.

<categories> satur vienu vai vairākus <category> elementus, <tags> - vienu vai vairākus <tags> elementus, un <terms> - vienu vai vairākus <term> elementus. Turpinājumā ir aprakstīts <category> elements (skat. 4.1. tabulu), bet <tag> un <term> elementu atribūti un apakšelementi ir gandrīz identiski, izņēmums ir <tag> elements, kurš nesatur parent atribūtu, jo tagi nav hierarhiska struktūra.

4.1. tabula

<category> elementa atribūti un apakšelementi

Nosaukums	Obligāts?	Apraksts	Piemērs
guid	jā	Globāli unikāls identifikators, sīkāks apraksts par guid atribūtiem WCML valodā ir 4.7. apakšnodaļā	<category guid=”kategorija_1”>
parent	nē	Guid norāde uz <categories> (<tags>, <terms>) elementā esošu apakšelementu.	<categories> <category guid=”kategorija_1”> <category guid="kategorija_2" parent="kategorija_1"> </categories>
<name>	jā	Kategorijas nosaukums	<name>Aktuālākie jaunumi</name>
<slug>	nē	Kategorijas aizstājvārds	<slug>aktualakie-jaunumi</slug>
<description>	nē	Kategorijas apraksts brīvā formā. Ieteicams aprakstu ievietot CDATA sekcijā.	<description><![CDATA[Kategorijas apraksts brīvā formā]]></description>

4.3. Izvēlnes

<menus> ir neobligāts <site> apakšelements, kas satur tīmekļa vietnes izvēlnes. Katra izvēlne ir iekļauta <menu> elementā. <menus> ir jāsatur viens vai vairāki <menu> elementi.

<menu> satur obligātu atribūtu menu-name. Šim atribūtam var piešķirt izvēlnes aizstājvārdu vai citu tekstu, bet tam jābūt unikālam starp visiem <menu> elementiem. <name> ir obligāts <menu> apakšelements, un satur izvēlnes nosaukumu. <description> ir neobligāts <menu> apakšelements, kas satur izvēlnes aprakstu brīvā tekstā. Šo elementa saturu ir ieteicams ievietot CDATA sekcijā. Izvēlne var saturēt vairākus izvēlnes ierakstus, kas tiek glabāti <menu-item> elementā. <menu-item> elementa atribūti un elementi ir apskatīti 4.2. tabulā.

4.2. tabula

<menu-item> elementa atribūti un apakšelementi

Nosaukums	Obligāts?	Apraksts	Piemērs
guid	jā	Globāli unikāls identifikators	<menu-item guid=" 05b9ea1b-76b8-4721 -9352-195d7ad19661">
parent	nē	Guid norāde uz <menu-item> elementu, kas atrodas tajā pašā <menu> elementā	<menu-item guid="menu_1" parent="menu_0">
state	nē	Izvēlnes elementa statuss[1]	<menu-item guid="izvelne_21" state="unpublished">
<title>	jā	Izvēlnes ieraksta virsraksts	<title>Aktualitātes </title>
<anchor-title>	nē	Apraksts, kas parādās, uzbraucot ar kursoru virs izvēlnes ieraksta (HTML title atribūts)	<anchor-title> Svaigākās ziņas </anchor-title>
<link>	nē	Izvēlnes ieraksta saite[2]	<link is-guid="false"> http://www.lu.lv/ </link>
<order>	nē	Ierakstu secība – skaitlis, pēc kura kārto izvēlnes elementus, ja elements nav norādīts, tad skaitli pieņem kā 0. Kārtošana notiek augošā secībā.	<order>14</order>

Tabulas 4.2 komentāri:

1) iespējamās vērtības: „published" (publicēts izvēlnes ieraksts), „unpublished„ (nepublicēts izvēlnes ieraksts). Ja atribūts nav norādīts, pēc noklusējuma vērtība tiek pieņemta kā „published";

2) <link> elementam ir neobligāts Būla atribūts is-guid, kas pēc noklusējuma ir true. Ja is-guid ir false, tad elementa vērtība ir URL adrese, citādi tā vērtība ir guid norāde uz jebkuru no elementiem, kuram ir guid atribūts.

4.4. Lietotāji

Šajā apakšnodaļā ir aprakstīts neobligāts <site> apakšelements <users>, kas satur vienu vai vairākus apakšelementus <user>. <user> elementi satur informāciju par vietnes lietotājiem. Sīkāka informācija par <user> atribūtiem un apakšelementiem ir norādīta 4.3. tabulā.

4.3. tabula

<user> elementa atribūti un apakšelementi

Nosaukums	Obligāts?	Apraksts	Piemērs
guid	jā	Globāli unikāls identifikators	<user guid="user_admin">
state	nē	Lietotāja statuss[1]	<user guid="user_andis" state="blocked">
<username>	jā	Unikāls lietotājvārds	<username>admin</username>
<name>	nē	Lietotāja vārds	<name>Andis</name>
<surname>	nē	Lietotāja uzvārds	<surname>Komarovs</surname>
<email>	nē	Lietotāja e-pasts	<email> andis.komarovs@gmail.com </email>
<website>	nē	Lietotāja mājas lapa	<website>http://www.lu.lv/ </websitie>
<description>	nē	Lietotāja biogrāfija/apraksts	<description> LU students </description>
<register-date>	nē	Lietotāja izveidošanas / reģistrācijas datums	<register-date> 2011-05-01T16:49:30+03:00 </register-date>

Tabulas 4.3 komentāri:

1) lietotāja statusa iespējamās vērtības ir „active" (aktīvs lietotāja profils) un „blocked" (bloķēts lietotāja profils), ja atribūts nav norādīts, tad tā vērtība ir „active".

4.5. Saturs

<content> ir neobligāts <site> apakšelements, kas satur vietnes tekstuālo saturu. Šajā WCML versijā ir definēts tikai viens satura veids – raksti, kas tiek attēlots, kā <content> apakšelementi <article> (skat 4.4. tabulu), bet nākošajās versijās WCML varētu saturēt arī tādus elementus, kā <poll> aptaujām, vai <banner> reklāmkarogiem. Visiem <content> apakšelementiem ir obligāts atribūts guid, kurš unikāli identificē saturu dokumenta ietvaros.

4.4. tabula

<article> elementa atribūti un apakšelementi

Nosaukums	Obligāts?	Apraksts	Piemērs
guid	jā	Globāli unikāls identifikators	<article guid="raksts_332">
state	nē	Raksta statuss[2]	<user guid="user_admin" state="blocked">
<title>	daļēji[1]	Virsraksts	<title>Virsraksts</title>
<alias>	nē	Aizstājvārds	<alias> raksta-aizstajvards</alias>
<summary>	nē	Raksta ievads HTML formātā, ir ieteicams šī elementa saturu likt CDATA sekcijā	<summary> <![CDATA[Raksta ievads]]> </summary>
<content>	daļēji[1]	Raksta saturs HTML formātā, ir ieteicams šī elementa saturu likt CDATA sekcijā	<content> <![CDATA[Raksta saturs]]> </content>
<author>	nē	Raksta autors. Šis ir teksta lauks, un ir paredzēts gadījumiem, kad lietotājs, kurš ir izveidojis rakstu nav raksta autors.	<author>Andis Komarovs</author>
<user>	nē	Raksta veidotājs. Šī ir atsauce uz <user> elementu ar norādīto guid atribūtu	<user>user_admin</user>
<created>	nē	Raksta izveidošanas datums	<created> 2011-04-13T11:31:26 </created>
<is-sticky>	nē	Būla pazīme par to, vai raksts ir īpaši izceļams, atkarībā no CMS to var interpretēt dažādi, piemēram, kā pazīmi par to, ka šis ir pirmās lapas raksts	<is-sticky>0</is-sticky>
<category>	nē	Raksta kategorija vai kategorijas, šo elementu var norādīt vairākas reizes. Šī ir atsauce uz <category> elementu.	<category> kategorija_2 </category>
<tag>	nē	Raksta tags vai tagi, šo elementu var norādīt vairākas reizes. Šī ir atsauce uz <tag> elementu.	<tag>tag_3</tag>
<comments>	nē	Raksta komentāri, šis elements ir	

		aprakstīts šīs apakšnodaļas turpinājumā	
<revisions>	nē	Raksta versijas, šis elements ir aprakstīts šīs apakšnodaļas turpinājumā	

Tabulas 4.4 komentāri:

1) obligāti jābūt vismaz vienam no elementiem <title> un <content>
2) raksta statusa iespējamās vērtības: „published" (publicēts raksts), „unpublished„ (nepublicēts raksts). Ja atribūts nav norādīts, tad tā vērtība tiek pieņemta kā „published".

4.5.1. Komentāri

<comments> ir neobligāts <article> apakšelements, bet šo elementu nepieciešamības gadījumā var izmantot arī citiem satura veidiem. <comments> elementā ietilpst viens vai vairāki <comment> elementi (skat. tabulu), kas satur komentārus. Tāpat <comments> ir neobligāts atribūts status, kura iespējamās vērtības ir „open" un „closed", kas attiecīgi nozīmē to, ka šim saturam ir atļauts vai aizliegts pievienot jaunus komentārus.

4.5. tabula

<comment> elementa atribūti un apakšelementi

Nosaukums	Obligāts?	Apraksts	Piemērs
guid	jā	Globāli unikāls identifikators	<comment guid="komentars_332_3">
state	nē	Komentāra statuss („published", „unpublished"), pēc noklusējuma „published"	<comment guid="komentars_332_3" state="published">
<name>	nē	Komentētāja vārds vai iesauka. Ieteicams norādīt vai nu <name>, vai <user>, bet nav aizliegts norādīt abus	<name>Andis K.</name>
<user>	nē	Reģistrēts lietotājs – komentāra autors	<user>user_admin</user>
<email>	nē	Komentētāja e-pasts	<email> andis.komarovs@gmail.com </email>
<website>	nē	Komentētāja mājas lapa	<website>www.lu.lv </website>
<subject>	nē	Komentāra temats, ieteicams to	<subject><![CDATA[Komentāra temats]]></subject>

		ievietot CDATA sekcijā	
<content>	jā	Komentāra teksts, ieteicams to ievietot CDATA sekcijā	<content><![CDATA[Komentāra teksts...]]></content>
<date>	nē	Komentāra izveidošanas laiks	<date>2011-04-13T15:31:26 </date>
<reply-to>	nē	Atbilde uz citu komentāru. Elements satur norādi uz citu <comment> elementu, kas atrodas tajā pašā <comments> elementā.	<reply-to> komentars_332_1 </reply-to>

4.5.2. Versijas

<revisions> elements satur raksta versijas. Tas ir neobligāts <article> apakšelements, un satur neobligātus <revision> elementus, kuriem ir obligāts atribūts nr, kas satur versijas kārtas numuru, un neobligāts atribūts date, kas satur versijas izveidošanas laiku.

<revision> satur elementus <title>, <alias>, <summary>, <content>, <author>, <user>, <created>, <is-sticky>, <category> un <tag>. Šiem elementiem ir tāda pati nozīme, kā <article> elementā, bet tie satur citas versijas datus.

4.6. Binārais saturs

Viena no problēmām, ar ko nākas saskarties migrācijas laikā ir neeksistējošs saturs. Saturu var sadalīt divās daļās – tekstuālais saturs, kas ir CMS pārvaldībā esošie dati, kas parasti atrodas datubāzē, un binārais saturs, jeb faili.

Migrējot datus visvieglākais veids, kā pārnest visus failus uz jauno sistēmu, ir pārkopēt tos uz jauno vietu, un tad par tiem aizmirst. Ja saites uz failiem ir norādītas relatīvi, tad šīs saites turpinās darboties, tomēr šāda pieeja nedod iespēju pārliecināties, vai visas saites, kas norādīja uz šiem failiem, joprojām darbojas. Iespējams kādai saitei ir norādīts absolūtais ceļš, citas savukārt var norādīt uz failu, kurš nezināmu iemeslu dēļ vairs neatrodas tam paredzētajā vietā. Tāpat saites var nenorādīt tieši uz failu, bet gan izmantot CMS iespējas failu lejupielādes nodrošināšanai, piemēram, pēc saites *www.piemers.lv/attachment?id=1* nav iespējams pateikt, kur atrodas lejupielādējamais fails, jo šī informācija tiek glabāta datubāzē.

Lai risinātu šo problēmu WCML piedāvā uzskaitīt failus, un gan eksporta, gan importa laikā pārbaudīt failu eksistenci. Uz uzskaitītajiem failiem ir iespējams atsaukties no teksta, izmantojot `wcml` vietturus, kas sīkāk aprakstīti 4.7 apakšnodaļā.

`<files>` elements ir neobligāts `<wcml>` apakšelements, kas satur `<file>` elementus. `<files>` elements nav `<site>` apakšelements, jo failu atrašanās vieta nav atkarīga no vietnes valodas vai vietņu skaita. `<file>` elementam ir divi obligāti atribūti: globāli unikāls identifikators - `guid`, un relatīvs ceļš no dokumenta līdz failam – `path`. WCML dokuments failus fiziski nesatur.

4.7. Saites

Lai risinātu iekšējo saišu problēmas, WCML valodā tiek izmantota dokumenta mēroga iekšējo atsauču sistēma. Atsauču sistēmu veido identifikatori un atsauces uz tiem. Visi identifikatori ir atribūti ar nosaukumu `guid`, un atribūtu tips ir `xsd:ID`. Uz `guid` atribūtiem var atsaukties trīs veidos:

1) izmantojot atribūtu ar tipu `xsd:IDREF`, piemēram `parent="vecaks"`;
2) izmantojot elementu, kurš satur vērtību ar tipu `xsd:IDREF`, piemēram `<parent>vecaks</parent>`;
3) izmantojot `wcml` vietturi tekstā, piemēram `{wcml://vecaks}`.

Pirmie divi veidi ir pašsaprotami, savukārt trešais veids ir paredzēts teksta saturā esošo saišu migrācijai. Šāds saišu migrācijas veids nepieciešams, jo saturs ne vienmēr ir korekts XHTML, bet ir nepieciešama metode, kā droši atpazīt un pārtulko `href`, `src` atribūtos vai vienkārši tekstā esošas saites. WCML tiek piedāvāts izmantot vietturi formā `{wcml://identifikators}`. Šāda forma ir izvēlēta, pirmkārt, tādēļ, ka neeksistē `wcml://` protokols, kas varētu radīt konfliktus, otrkārt, tādēļ, ka `xsd:ID` atribūts nedrīkst saturēt figūriekavas. Šāda kombinācija ļauj ar vienkāršu regulāro izteiksmi atlasīt visus vietturus, un nomainīt tos ar iztulkotām adresēm. Regulārās izteiksme (PCRE [25] sintaksē), kas atrod doto vietturi ir `#(\{wcml://([^\}]+)\})#`. Par korektu vietturu ievietošanu un to tulkošanu ir atbildīgs valodas interpretators. Vietturus drīkst izmantot šādos elementos: `<description>`, `<summary>` un `<content>`.

4.8. WCML paplašināšana

WCML piedāvā tikai CMS datu pamatstruktūras, un tādēļ valodas pielietojums ir ierobežots. Šī iemesla dēļ WCML ir paredzēta iespēja valodu paplašināt ar XML vārdtelpu [23] palīdzību. Noteikumi ir diezgan vienkārši – WCML drīkst saturēt jebkādus šajā nodaļā

neparedzētus elementus un atribūtus, ja vien tie ir definēti atsevišķā vārdtelpā, un neizjauc esošo elementu struktūru. Papildus tam ir ieteicams izmantot esošos elementus datu grupēšanai, piemēram `<terms>` elementu WCML nedefinētiem taksoniem, vai satura glabāšanu `<content>` elementā.

Izmantojot paplašinātu WCML ir iespējams migrēt specifiskākus datus, bet tas prasa gan eksporta, gan importa procedūru pielāgojumus.

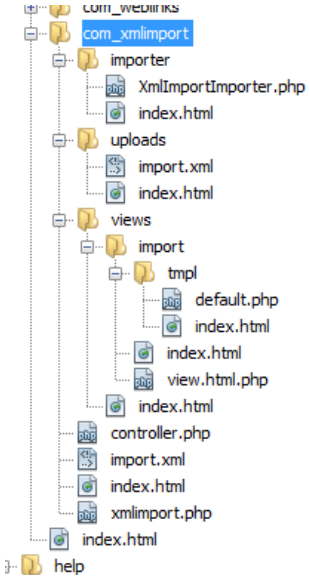
4.9. Datu importa piemērs

Šajā apakšnodaļā ir apskatīts neliels datu importa piemērs. Šī piemēra mērķis ir praktiski pārliecināties par to, ka WCML dokumentu ir iespējams ieimportēt reālā satura pārvaldības sistēmā. Datu eksports šajā darbā nav apskatīts, jo tas no programmatūras koda viedokļa ir līdzīgs.

Piemērs ir realizēts Joomla satura pārvaldības sistēmā, kā dati ir izmantoti divi rakstu fragmenti no adresēm http://www.df.lu.lv/par/kontaktinformacija/ un http://www.df.lu.lv/par/koncepcija/, rakstos ir ievietotas saites, kas norāda uz otru rakstu, tādejādi izveidojot saišu ciklu, tāpat vienā rakstā ir ievietots Latvijas Universitātes logo. Logo attēla fails testa nolūkiem ir uzkopēts lokāli uz servera, Joomla saknes mapē. WCML dokumenta piemērs ir dots 2. pielikumā.

Komponenta izveide un imports

Importa nolūkiem ir izveidots pielāgots komponents `com_xmlimport`, kas veidots pēc Joomla dokumentācijā esošas pamācības [26], *4.1.* attēlā ir norādīta komponenta struktūra administrācijas daļā.



4.1. att. **com_xmlimport komponenta struktūra**

No norādītās struktūras vissvarīgākais ir `XmlImportImporter.php` fails, kas arī ir pievienots 3. pielikumā. Imports tiek palaists izsaucot šajā failā esošo `XmlImportImporter` klases funkciju `import`, kas kā parametru saņem WCML dokumentu teksta virknes formā.

Tā kā šis ir demonstrācijas piemērs, `import` funkcijas algoritms ir ļoti vienkāršs:

1) ar XPath izteiksmi „`/wcml/site/content/article`" atlasa visus rakstus;
2) ieimportē rakstus, kategorijas un lietotājus;
3) veic saišu tulkošanu.

Demonstrācijas nolūkos, imports nepārbauda vai datubāzē jau eksistē tādi paši lietotāji vai kategorijas. Raksti, kategorijas un lietotāji tiek ievietoti ar Joomla klases `JTable` palīdzību, kas automātiski nodrošina korektu tiesību ierakstu izveidošanu, kā arī ļauj veikt datu kvalitātes pārbaudes.

Saišu tulkošana tiek veikta ar regulāro izteiksmju palīdzību. Tiek sameklēts `wcml` vietturis, no tā iegūts `guid` identifikators, pēc `guid` identifikatora tiek sameklēts attiecīgais elements, un atkarībā no elementa nosaukuma tiek izveidota saite vai nu uz rakstu, vai kategoriju vai failu.

Rezultāts

Datu importa rezultāts ir redzams attēlos *4.2* un *4.3*. Kā redzams, ir ieimportēti abi raksti, tiek korekti attēlots fails – Latvijas Universitātes logo, ir korekti uzstādīta kategorija, kā arī kategorijas skatā tiek korekti attēloti autori (`<author>Latvijas Universitāte </author>`, un `<author>Andis Komarovs</author>`).

4.2.att. **Importēto rakstu kategorijas skats**

4.3. att. **Importētais raksts „Koncepcija – pirmā rindkopa" pilnā atvērumā**

SECINĀJUMI

Šī darba galvenais mērķis bija izveidot vienotu datu formātu, kas atvieglotu datu migrāciju starp tīmekļa satura pārvaldības sistēmām. Šis mērķis ir sasniegts – ir izveidota tīmekļa satura iezīmēšanas valodas (WCML) versija 0.5, kas iekļauj populārāko tīmekļa satura pārvaldības sistēmu datu struktūras, paredz valodas paplašināšanas iespējas, un sniedz atbalstu tipisku migrācijas problēmu risināšanā. Tāpat ir izveidots neliels importa piemērs, kas parāda formāta dzīvotspēju reālos apstākļos.

Jāapzinās, ka satura migrācija nav triviāls uzdevums. Pat ja ir iespējama automatizēta satura migrācija, tik un tā būs nepieciešams darbs, lai jauno sistēmu nokonfigurētu, apstrādātu ieimportēto saturu, un veiktu migrācijas testēšanu. Turklāt, lai veiksmīgi migrētu datus, abās sistēmās ir jābūt līdzīgai funkcionalitātei, jo, piemēram, nav iespējams migrēt aptauju datus uz sistēmu, kura aptaujas nepiedāvā.

Teorētiski, izmantojot WCML, ir iespējams panākt automatizētu visa satura migrāciju gan no, gan uz visām satura pārvaldības sistēmām, kuras atbalsta WCML. Tomēr praktiski šāds scenārijs šobrīd ir iespējams tikai ar vienkāršiem datiem, piemēram, tīmekļa dienasgrāmatām, cita veida datu migrācijai ir jāveic WCML pielāgošana. Šāds rezultāts ir likumsakarīgs un prognozēts, jo darbā tika apskatītas tikai CMS standarta instalācijas.

Veicot CMS datu apkopošanu, kļuva skaidrs, kādēļ neeksistē meklētais datu apmaiņas formāta standarts. Satura pārvaldības sistēmu apgabals ir pārāk plašs un izplūdis. Šajā darbā ir izvēlēta tikai daļa no visa CMS apgabala, bet tik un tā nevar novilkt skaidru robežu starp datiem, kurus vispārīgā gadījumā ir nepieciešams migrēt, un kurus nav. Savukārt datu apgabalam ir nozīme veidojot iezīmēšanas valodas struktūru, jo ir divas galējības – neiekļaut nevienu elementu vai iekļaut visus datu apgabala elementus. Neiekļaujot nevienu tiek iegūts „pliks" XML, savukārt pat, ja izdodas iekļaut visus elementus no plaša datu apgabala, kuram nav skaidras robežas, valoda sanāk pārāk apjomīga praktiskai lietošanai. Šajā jomā veiksmīga kompromisa piemērs, šķiet ir RSS, jo tas satur tikai nelielu daļu no tīmekļa informācijas, tomēr šī daļa ir praktiski izmantojama, un RSS savas vienkāršības dēļ, ir ieguvis lielu popularitāti.

Kā jau minēts 4. nodaļā, autors WCML neuzskata par pilnībā pabeigtu, jo uzskata, ka pētījumā vajadzētu iekļaut arī populārākos CMS paplašinājumus, bet ņemot vērā paplašinājumu skaitu, tas šajā darbā nav darīts. Šī situācija ļauj nākotnē turpināt darba tēmu, izpētot lielāku datu apjomu, tomēr ir svarīgi laikus apstāties, lai WCML nekļūtu par pārāk sarežģītu valodu.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. **RSS Advisory Board.** RSS 2.0 Specification. [Tiešsaiste] 2009. gada 30. marts. [Citēts: 2011. gada 18. maijs.] http://www.rssboard.org/rss-specification.

2. **Network Working Group.** The Atom Syndication Format. [Tiešsaiste] 2005. gada decembris. [Citēts: 2011. gada 18. maijs.] http://tools.ietf.org/html/rfc4287.

3. **Robertson, James.** Content migration: options and strategies. *Step Two Designs.* [Tiešsaiste] 2008. gada 21. Jūnijs. [Citēts: 2011. gada 8. Maijs.] http://www.steptwo.com.au/papers/kmc_migration/index.html.

4. **kapow software.** *Automating Content Migration The Definitive Guide.*

5. **Hillyer, Mike.** Managing Hierarchical Data in MySQL. *MySQL.* [Tiešsaiste] [Citēts: 2011. gada 13. maijs.] http://dev.mysql.com/tech-resources/articles/hierarchical-data.html.

6. **Vamosa.** Content Migration Seven Steps to Success. *Vamosa.* [Tiešsaiste] [Citēts: 2011. gada 13. maijs.] http://www.vamosa.com/content-migration-seven-steps-to-success-d49.

7. **Yann Rocq.** Wordpress Import. *Drupal.* [Tiešsaiste] 2007. gada 17. decembris. [Citēts: 2011. gada 14. maijs.] http://drupal.org/project/wordpress_import.

8. **Gupta, Arun.** Language-neutral data format: XML and JSON. *Java.net The Source for Java Technology Collaboration.* [Tiešsaiste] 2007. gada 1. marts. [Citēts: 2011. gada 18. maijs.] http://weblogs.java.net/blog/arungupta/archive/2007/03/languageneutral.html.

9. **Goessner, Stefan.** Converting Between XML and JSON. *O'reilly xml from the inside out.* [Tiešsaiste] 2006. gada 31. maijs. [Citēts: 2011. gada 18. maijs.] http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html.

10. **Marinescu, Floyd un Tilkov, Stefan.** Debate: JSON vs. XML as a data interchange format. *InfoQ.* [Tiešsaiste] 2006. gada 26. decembris. [Citēts: 2011. gada 18. maijs.] http://www.infoq.com/news/2006/12/json-vs-xml-debate.

11. **Wordpress.** Importing Content. *WordPress.org.* [Tiešsaiste] [Citēts: 2011. gada 18. maijs.] http://codex.wordpress.org/Importing_Content.

12. **OASIS.** Content Management Interoperability Services (CMIS) Version 1.0. [Tiešsaiste] 2010. gada 1. maijs. [Citēts: 2011. gada 18. maijs.] http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.pdf.

13. **water & stone.** *2010 open source CMS market share report.* 2010.

14. *Drupal.* [Tiešsaiste] [Citēts: 2011. gada 13. maijs.] http://drupal.org/.

15. *Joomla!* [Tiešsaiste] [Citēts: 2011. gada 13. maijs.] http://www.joomla.org/.

16. *WordPress.org.* [Tiešsaiste] [Citēts: 2011. gada 13. maijs.] http://wordpress.org/.

17. **WordPress.com.** Categories vs. Tags. [Tiešsaiste] 2010. gada 21. oktobris. [Citēts: 2011. gada 21. maijs.] http://en.support.wordpress.com/posts/categories-vs-tags/.
18. **Encyclopædia Britannica.** markup language. *Encyclopædia Britannica Online.* [Tiešsaiste] 2011. gada. [Citēts: 2011. gada 19. maijs.] http://www.britannica.com/EBchecked/topic/1323641/markup-language.
19. **Boiko, Bob.** What Are Content Markup Languages? [Tiešsaiste] 2002. gada. [Citēts: 2011. gada 19. maijs.] http://metatorial.com/downloads/Boiko_Wp_WhatAreContentMarkupLanguages.pdf.
20. **Bray, Tim.** On Semantics and Markup. [Tiešsaiste] 2003. gada 9. aprīlis. [Citēts: 2011. gada 19. maijs.] http://www.tbray.org/ongoing/When/200x/2003/04/09/SemanticMarkup.
21. **Refsnes Data.** XML Tutorial. *w3schools.com.* [Tiešsaiste] [Citēts: 2011. gada 20. maijs.] http://www.w3schools.com/xml/default.asp.
22. **World Wide Web Consortium.** Extensible Markup Language (XML) 1.0 (Fifth Edition). [Tiešsaiste] 2008. gada 26. novembris. [Citēts: 2011. gada 20. maijs.] http://www.w3.org/TR/REC-xml/.
23. —. Namespaces in XML 1.0 (Third Edition). [Tiešsaiste] 2009. gada 8. decembris. [Citēts: 2011. gada 20. maijs.] http://www.w3.org/TR/xml-names/.
24. **Vlist, Eric van der.** The Best Way to Validate XML Document Structures. [Tiešsaiste] 2003. gada. [Citēts: 2011. gada 20. maijs.] http://books.xmlschemata.org/relaxng/relax-CHP-1-SECT-4.html.
25. **Hazel, Philip.** *PCRE - Perl Compatible Regular Expressions.* [Tiešsaiste] [Citēts: 2011. gada 18. maijs.] http://www.pcre.org/.
26. **Joomla.** Developing a Model-View-Controller (MVC) Component for Joomla!1.6. *Joomla! Documentation.* [Tiešsaiste] 2011. gada 16. janvāris. [Citēts: 2011. gada 18. maijs.] http://docs.joomla.org/Developing_a_Model-View-Controller_%28MVC%29_Component_for_Joomla!1.6.

PIELIKUMI

1. pielikums

WCML dokumenta paraugs ar visiem elementiem un atribūtiem

```
<?xml version="1.0" encoding="utf-8"?>
<wcml version="0.5" xmlns:xml="http://www.w3.org/XML/1998/namespace">
    <site xml:lang="lv-LV">
        <base-url>http://example.localhost/</base-url>
        <taxonomy>
            <categories>
                <category guid="kategorija_1">
                    <name>Jaunumi</name>
                    <slug>jaunumi</slug>
                    <description><![CDATA[Jaunumu
kategorija]]></description>
                </category>
                <category guid="kategorija_2" parent="kategorija_1">
                    <name>IT Jaunumi</name>
                    <slug>it-jaunumi</slug>
                    <description><![CDATA[IT jaunumu
kategorija]]></description>
                </category>
            </categories>
            <tags>
                <tag guid="tag_1">
                    <name>PHP</name>
                    <slug>php</slug>
                    <description><![CDATA[Viss, kas saistīts ar
PHP]]></description>
                </tag>
                <tag guid="tag_2">
                    <name>.NET</name>
                    <slug>net</slug>
                </tag>
                <tag guid="tag_3">
                    <name>Java</name>
                    <slug>java</slug>
                </tag>
            </tags>
            <terms name="links">
                <term guid="saisu_kategorija_1">
                    <name>Partneri</name>
                    <slug>partneri</slug>
                    <description><![CDATA[Saišu kategorija, kas satur
partneru mājas lapu adreses]]></description>
                </term>
                <term guid="saisu_kategorija_2"
parent="saisu_kategorija_1">
                    <name>Zelta partneri</name>
                    <slug>zelta-partneri</slug>
                    <description><![CDATA[Saišu kategorija, kas satur zelta
partneru mājas lapu adreses]]></description>
                </term>
            </terms>
            <terms name="polls">
                <term guid="aptauju_kategorija_1">
                    <name>IT aptaujas</name>
                    <description><![CDATA[Aptaujas, kas saistītas ar IT
jautājumiem]]></description>
```

```
                </term>
                <term guid="aptauju_kategorija_2">
                    <name>Citas aptaujas</name>
                </term>
            </terms>
        </taxonomy>
        <menus>
            <menu menu-name="mainmenu">
                <name>Galvenā izvēlne</name>
                <description><![CDATA[Vietnes galvenā
izvēlne]]></description>
                <menu-item guid="menu_1" state="published">
                    <title>Aktualitātes</title>
                    <anchor-title>Aktuālākie jaunumi</anchor-title>
                    <link>kategorija_1</link>
                    <order>1</order>
                </menu-item>
                <menu-item guid="menu_2" parent="menu_1"
state="unpublished">
                    <title>Izvēlnes ieraksts 2</title>
                    <link is-guid="false">http://www.cita-
majaslapa.lv</link>
                    <order>1</order>
                </menu-item>
            </menu>
            <menu menu-name="sidemenu">
                <name>Sānu izvēlne</name>
                <menu-item guid="sidemenu_1" state="unpublished">
                    <title>Izvēlnes ieraksts</title>
                    <link>raksts_332</link>
                    <order>1</order>
                </menu-item>
            </menu>
        </menus>
        <users>
            <user guid="user_admin" state="active">
                <username>admin</username>
                <name>Andis</name>
                <surname>Komarovs</surname>
                <email>andis.komarovs@gmail.com</email>
                <website>www.lu.lv</website>
                <description><![CDATA[LU students]]></description>
                <register-date>2011-04-01T20:13:16</register-date>
            </user>
        </users>
        <content>
            <article guid="raksts_332" state="published">
                <title>Virsraksts</title>
                <alias>virsraksts</alias>
                <summary><![CDATA[Raksta ievads]]></summary>
                <content><![CDATA[Raksta saturs ' & <<<]]></content>
                <author>Andis Komarovs</author>
                <user>user_admin</user>
                <created>2011-04-13T11:31:26</created>
                <is-sticky>0</is-sticky>
                <category>kategorija_2</category>
                <tag>tags_2</tag>
                <tag>tags_3</tag>
                <comments status="open">
                    <comment guid="komentars_332_1" state="published">
                        <name>Andis K.</name>
                        <user>user_admin</user>
                        <email>andis.komarovs@gmail.com</email>
```

```
                        <website>www.lu.lv</website>
                        <subject><![CDATA[Komentāra temats]]></subject>
                        <content><![CDATA[Komentāra teksts...]]></content>
                        <date>2011-04-13T15:31:26</date>
                        <reply-to />
                    </comment>
                </comments>
                <revisions>
                    <revision nr="1" date="2011-04-13T10:35:26">
                        <title>Virsraksts</title>
                        <alias>virsraksts</alias>
                        <summary><![CDATA[Raksta ievads]]></summary>
                        <content><![CDATA[<a href="{wcml://file1}">Raksta
saturs</a>]]></content>
                        <author>Andis Komarovs</author>
                        <user>user_admin</user>
                        <created>2011-04-13T10:31:26</created>
                        <is-sticky>1</is-sticky>
                        <category>kategorija_2</category>
                        <tag>tags_1</tag>
                    </revision>
                </revisions>
            </article>
        </content>
    </site>
    <files>
        <file guid="file1" path="images/logo.png" />
        <file guid="file2" path="files/data.pdf" />
    </files>
</wcml>
```

Joomla importa piemērs – WCML dokuments

```
<?xml version="1.0" encoding="utf-8"?>
<wcml version="0.5" xmlns:xml="http://www.w3.org/XML/1998/namespace">
    <site xml:lang="lv-LV">
        <base-url>http://example.localhost/</base-url>
        <taxonomy>
            <categories>
                <category guid="category_1">
                    <name>LU</name>
                    <slug>lu</slug>
                </category>
            </categories>
        </taxonomy>
        <users>
            <user guid="andis" state="active">
                <username>andis</username>
                <name>Andis</name>
                <surname>Komarovs</surname>
                <email>andis.komarovs@gmail.com</email>
                <website>www.lu.lv</website>
                <description><![CDATA[LU students]]></description>
                <register-date>2011-04-01T20:13:16</register-date>
            </user>
        </users>
        <content>
            <article guid="article_7" state="published">
                <title>LU DF Kontaktinformācija</title>
                <content><![CDATA[<strong>Adrese</strong>: Raiņa bulvāris
19, Rīga, LV 1586
<strong>Kā  atrast:</strong> pilsētas centrā, LU galvenā ēka
<strong>Tālrunis</strong>:  67034488
<strong>Dekāns:</strong> profesors Juris Borzovs <strong>
Fakultātes  e-pasts</strong>: <a href="mailto:df@lu.lv">df@lu.lv</a>
<strong>Mājas lapa:</strong> <a href="http://www.df.lu.lv/df/"
target="_blank">www.df.lu.lv</a>

<a title="Koncepcija - pirmā rindkopa" href="{wcml://article_9}">LU DF
Koncepcija</a>]]>
                </content>
                <author>Andis Komarovs</author>
                <user>andis</user>
                <created>2011-05-16 20:19:15</created>
                <is-sticky>0</is-sticky>
                <category>category_1</category>
            </article>
            <article guid="article_9" state="active">
                <title>Koncepcija - pirmā rindkopa</title>
                <alias>koncepcija-pirma-rindkopa</alias>
                <summary><![CDATA[Pēdējos 10 gados nozare, kurā ietilpst
datorzinātne, informācijas  tehnoloģijas un komunikāciju tehnoloģijas, ir
piedzīvojusi strauju  izaugsmi gan pasaulē, gan Latvijā. Nozare pēc
Ekonomikas ministrijas  datiem ir saražojusi apmēram 5-6 % no Latvijas IKP;
gandrīz 100 miljonos  ir mērāms eksporta apjoms.]]>
                </summary>
                <content><![CDATA[Valdības deklarācijās atkārtoti nozare ir
pasludināta par valsts prioritāti. Diemžēl norādītajai nozarei latviešu
valodā nav vienota nosaukuma. Bieži tiek lietoti termini "Datorzinātne",
"Informātika" "Informācijas tehnoloģijas" vai "Informācijas un
komunikāciju tehnoloģijas". Dotajā dokumentā, atbilstoši Izglītības un
zinātnes ministrijas 2004.gada 11.maijā apstiprinātajam (Rīkojums Nr.  287)
izglītības klasifikatoram, tiek lietots vārds <em>"Datorika"</em> kā
```

```
angliskā termina <em>“Computing”</em> latviskā atbilsme. Kā zināms, termins
<em>“Computing” </em>tiek  lietots, lai apzīmētu nozari, kurā ietilpst šādi
virzieni: datorzinātne  (computer science), informācijas tehnoloģijas
(information  technologies), informācijas sistēmas (information systems),
programmatūras inženierija (software engineering) un datoru inženierija
(computer engineering).

<a href="{wcml://file_1}"><img class="alignnone size-full wp-image-16"
title="LU logo" src="{wcml://file_1}" alt="" width="253" height="124"
/></a>

<a title="LU DF Kontaktinformācija"
href="{wcml://article_7}">Kontaktinformācija</a>]]>
                </content>
                <author>Latvijas Universitāte</author>
                <user>andis</user>
                <created>2011-05-16 20:19:28</created>
                <is-sticky>0</is-sticky>
                <category>category_1</category>
            </article>
        </content>
    </site>
    <files>
        <file guid="file_1" path="/wp-content/uploads/2011/05/LU-logo.jpg"
/>
    </files>
     </wcml>
```

Joomla importa piemērs – XmlImportImporter.php fails

```php
<?php

// No direct access to this file
defined('_JEXEC') or die('Restricted access');

class XmlImportImporter {

    private $categories = array();
    private $users = array();
    private $articles = array();
    private $wcml;

    public function import($xmlString) {
        $wcml = simplexml_load_string($xmlString, 'SimpleXMLElement',
LIBXML_NOCDATA);
        $this->wcml = $wcml;
        foreach ($wcml->xpath('/wcml/site/content/article') as $item) {
            $attributes = $item->attributes();
            $data = array();
            $data['title'] = (string) $item->title;
            $data['alias'] = (string) $item->alias;
            if ($item->summary) {
                $data['introtext'] = (string) $item->summary;
                $data['fulltext'] = (string) $item->content;
            } else {
                $data['introtext'] = (string) $item->content;
            }
            $data['created_by_alias'] = (string) $item->author;
            $data['created'] = (string) $item->created;
            switch ($item->state) {
                case 'published':
                    $data['state'] = 1;
                    break;
                case 'unpublished':
                    $data['state'] = 0;
                    break;
                default:
                    $data['state'] = 1;
                    break;
            }
            $data['featured'] = filter_var($item->{"is-sticky"},
FILTER_VALIDATE_BOOLEAN);
            if (count($item->category) > 1) {
                echo 'warning - more than one category (guid="' .
$attributes['guid'] . '")';
            }

            //process categories
            if ($item->category) {
                $category = $wcml->xpath('//category[@guid="' . $item-
>category . '"]');
                $category = $category[0];
                if (!$category) {
                    echo 'fatal error - no category with guid="' . $item-
>category . '"';
                    return;
                }
                $catData = array();
                $catData['title'] = (string) $category->name;
                $catData['alias'] = (string) $category->slug;
```

```
                $catData['description'] = (string) $category->description;
                $categoryDb = $this->saveCategory((string) $item->category,
$catData);
                $data['catid'] = $categoryDb->id;
            }

            //process user
            if ($item->user) {
                $user = $wcml-
>xpath('/wcml/site/users/user/username[normalize-space()="' . $item->user .
'"]/..');
                $user = $user[0];
                if (!$user) {
                    echo 'fatal error - no user with username="' . $item-
>user . '"';
                    return;
                }
                $usrData = array();
                $usrData['username'] = (string) $user->username;
                $usrData['name'] = trim($user->name . ' ' . $user-
>surname);
                $usrData['email'] = (string) $user->email;
                $usrData['registerDate'] = (string) $user->{"register-
date"};
                $usrAttr = $user->attributes();
                switch ($usrAttr['state']) {
                    case 'active':
                        $usrData['block'] = 0;
                        break;
                    case 'blocked':
                        $usrData['block'] = 1;
                        break;
                    default:
                        $usrData['block'] = 0;
                        break;
                }
                $userDb = $this->saveUser($usrData);
                $data['created_by'] = $userDb->id;
            }

            $articleDb = $this->saveArticle((string)$attributes['guid'],
$data);
            if ($data['featured']) {
                $sql = "INSERT INTO `jos_content_frontpage` (`content_id`)
VALUES (" . $articleDb->id . ")";
                $db = JFactory::getDbo();
                $db->setQuery($sql);
                $db->query();
                if ($db->getErrorMsg()) {
                    JError::raiseWarning(500, $db->getErrorMsg());
                }
            }
        }
        $this->translateLinks();
    }

    protected function translateLinks(){
        $pattern = '#(\{wcml://([^\}]+)\})#';
        foreach ($this->articles as $article) {
            $article->introtext = $this->pregReplaceLinks($pattern,
$article->introtext);
            $article->fulltext = $this->pregReplaceLinks($pattern,
$article->fulltext);
```

```
            if (!$article->store()) {
                JError::raiseError(500, $article->getError());
            }
        }
    }

    private function pregReplaceLinks($pattern, $subject){
        preg_match_all($pattern, $subject, $matches);
        $replacements = array();
        foreach ($matches[1] as $key=>$match) {
            $replacements[$matches[0][$key]] = $this-
>getLink($matches[2][$key]);
        }
        return str_replace(array_keys($replacements),
array_values($replacements), $subject);
    }

    protected function getLink($guid){
        $node = $this->wcml->xpath('//*[@guid="'.$guid.'"]');
        $node = $node[0];
        $name = $node->getName();
        $link = "";
        switch ($name) {
            case 'article':
                $article = $this->articles[$guid];
                $link =
'index.php?option=com_content&view=article&id='.$article->id.':'.$article-
>alias.'&catid='.$article->catid;
                break;
            case 'file':
                $attributes = $node->attributes();
                $path = $attributes['path'];
                $path = ltrim($path, '/\\');
                $link = JURI::root().$path;
                break;
            case 'category':
                $category = $this->categories[$guid];
                $link =
'index.php?option=com_content&view=category&id='.$category->id;
                break;
            default:
                JError::raiseError(500, "Not supported guid element
type!");
                break;
        }
        return $link;
    }

    protected function saveArticle($guid, $data) {
        if (!isset($this->articles[$guid])) {
            $row = & JTable::getInstance('content');
            if (!$row->bind($data)) {
                return JError::raiseWarning(500, $row->getError());
            }
            if (!$row->check()) {
                JError::raiseError(500, $row->getError());
            }
            if (!$row->store()) {
                JError::raiseError(500, $row->getError());
            }
            $this->articles[$guid] = $row;
        }
```

```
            return $this->articles[$guid];
        }

        protected function saveCategory($guid, $data) {
            if (!isset($this->categories[$guid])) {
                $row = & JTable::getInstance('category');
                $row->extension = 'com_content';
                $row->setLocation(1, 'last-child');
                $row->published = 1;
                if (!$row->bind($data)) {
                    return JError::raiseWarning(500, $row->getError());
                }
                if (!$row->check()) {
                    JError::raiseError(500, $row->getError());
                }
                if (!$row->store()) {
                    JError::raiseError(500, $row->getError());
                }
                $this->categories[$guid] = $row;
            }
            return $this->categories[$guid];
        }

        protected function saveUser($data) {
            if (!isset($this->users[($data['username'])])) {
                $row = & JTable::getInstance('user');
                $row->password =
"67a6eecf087d2408de23ecaa7f787871:iyomwqn7EPjHglMb9eENpvfAprkYazc0";
//default password: asdf12
                if (!$row->bind($data)) {
                    return JError::raiseWarning(500, $row->getError());
                }
                if (!$row->check()) {
                    JError::raiseError(500, $row->getError());
                }
                if (!$row->store()) {
                    JError::raiseError(500, $row->getError());
                }
                $this->users[($data['username'])] = $row;
            }
            return $this->users[($data['username'])];
        }
         }
```

Maģistra darbs: **Migrācijas valoda satura pārvaldības sistēmām**

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **piemērotu / nepiemērotu** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____
(Vadītāja paraksts)

Darbs iesniegts **maģistrantūras sekretariātā** _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents:

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____ , vērtējums _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)