hyväksymispäivä arvosana

arvostelija

XML-metakielen hyödyntäminen oppimateriaalin tuottamisessa

Ahti Syreeni

Helsinki 10.1.2005

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Ahti Syreeni

Työn nimi — Arbetets titel — Title

XML-metakielen hyödyntäminen oppimateriaalin tuottamisessa

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
LuK-tutkielma	10.1.2005	20 sivua

Tiivistelmä — Referat — Abstract

XML-metakielellä voidaan kuvata dokumentin sisältö ja samantyyppisille XML-dokumentin ilmentymille voidaan kirjoittaa muunnossäännöt XSLT-kieltä käyttäen. XSLT-tulkin avulla XML-dokumenttityypin ilmentymästä ja muunnossäännöistä saadaan dokumentin esitys. Tätä tekniikkaa, jossa sisältö ja sen esitys erotetaan, voidaan käyttää myös oppimateriaalin tuottamiseen.

XML-metakieltä voidaan myös käyttää lisättäessä valmiisiin oppimateriaaleihin metatietoa. Oppimateriaaliin lisättävän metatiedon rakenteesta on olemassa standardeja ja standardiluonnoksia. Tällä hetkellä vallitseva oppimateriaalin metatiedon standardi on IEEE LOM.

Kokonaisten opetusaihealueiden ja kurssien oppimateriaalin sisällön kuvaaminen XML-metakielellä mahdollistaa oppimateriaalin käytön uudelleen, oppimateriaalin tehokkaan hallinnan ja vaihtuvat esitysmuodot.

Learning Material Markup Language on esimerkki opetusaihealueiden kuvaamisessa käytettävistä XML-pohjaisista kielistä.

ACM Computing Classification System (CCS):
I.7.4 [Electronic Publishing],
H.5.4 [Hypertext/Hypermedia]

Avainsanat — Nyckelord — Keywords

XML-metakieli, oppimateriaali

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Tietotekniikan kehittyminen ja tietoliikenneyhteyksien paraneminen ovat mahdollistaneet erilaisten sähköisten apuvälineiden käytön opetuksessa. Tietoteknisen kehityksen huipentumana ovat tietokoneavusteiset oppimisympäristöt, joissa parhaimmillaan opetusmateriaalin esitykseen vaikuttavat oppijan esitiedot ja tausta.

Lisääntyneet mahdollisuudet asettavat vaatimuksia opetuksessa käytettävälle oppimateriaalille. Sen tulisi olla käytettävissä monissa eri järjestelmissä ja esitysmuodoissa sekä tulostettavissa paperiversioiksi. Oppimateriaalin tulisi olla entistä helpommin löydettävissä, käytettävissä uudelleen ja tuotettavissa tehokkaasti. Vaatimukset ovat ohjanneet materiaalin tuottajat etsimään ratkaisua oppimateriaalin sisällön kuvaamisesta metatiedon avulla.

Metatieto on tietoa tiedosta. Se kertoo tiedon sellaisista ominaisuuksista, joita tiedosta ei koneellisesti voida tehokkaasti laskea. Metatieto ei olekaan tarkoitettu ihmisille, vaan koneille, jotka tietoa käsittelevät. Metatiedon lisääminen dokumentteihin mahdollistaa niiden tehokkaan koneellisen käsittelyn.

Jotta metatietoa voitaisiin lisätä dokumentteihin, on sovittava tapa, jolla metatieto voidaan erottaa muusta tiedosta. Sopimus merkintätavasta on metakieli. Merkintätavalla on säännöt, joiden mukaisesti dokumentteja käsittelevät ohjelmat voidaan ohjelmoida. Merkintätavalla on tietty rakenne ja ominaisuudet, metakielen syntaksi.

SGML (Standard Generalized Markup Language) ja XML (Extensible Markup Language) ovat käytetyimpiä metakieliä. ISO (International Organization for Standardization) standardoi SGML:n vuonna 1986 ja W3C (World Wide Web Consortium) kehitti SGML:stä paremmin www-ympäristöön sopivan, suppeamman version, XML-metakielen. XML-metakielen ensimmäinen versio ilmestyi vuonna 1998.

Tässä tutkielmassa esitellään XML-metakielen käyttötapoja oppimateriaalin tuottamisessa. Ensin luvussa 2 luodaan katsaus XML-metakielen perusominaisuuksiin ja samalla selviää, miten yksittäisiä opetusdokumentteja voidaan tuottaa XML- metakielen avulla. Tämän jälkeen luvussa 3.1 tutustutaan metatiedon liittämiseen valmiisiin oppimateriaaleihin. Metatiedon liittämisen ja oppimateriaalin tuottamisen ideat yhdistyvät oppimateriaalin tuottamista XML-metakielen avulla käsittelevässä luvussa 3.2. Tämän jälkeen on pohdittu molempien tapojen etuja sekä haittoja. Lopuksi luvussa 4 on esitelty esimerkkinä opetusmateriaalin tuottamiseen kehitetyistä XML-sovelluksista LMML-kieli, jonka avulla kuvataan kokonaisia opetusaihealueita.

2 XML-metakieli

XML-metakielen käyttö on tapa erottaa dokumenttien sisältö ja esitystapa toisistaan. Dokumentin sisältö merkataan XML-metakielellä, jonka rakenne määritetään rakennemäärittelyssä. Samaa rakennemäärittelyä noudattaville eli saman tyyppisille dokumenteille tuotetaan esitys tyylitiedostoa käyttäen.

2.1 Sisällön merkkaus

XML-metakielessä dokumentin sisältö merkataan elementeillä. Elementti on erotettu ympäröivästä tekstistä kulmasulkein ja jokaisella elementillä on vastinelementti. Vastinelementti merkitään kauttaviivalla [B+00]: XML-merkatussa lauseessa `<nimi>John von Neumann</nimi> syntyi <syntymaaika>28.12.1903</syntymaaika>` elementille `<nimi>` on vastinelementti `</nimi>`. Väliin jäävä teksti `John von Neumann` on näin merkattu: siihen on liitetty metatieto tekstin tarkoituksesta. Sama pätee elementille `<syntymaaika>`.

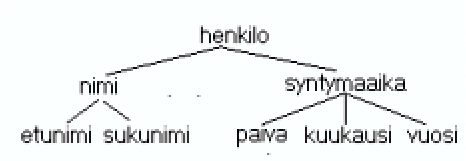
Elementit voivat olla sisäkkäisiä. Tällöin elementin ja sen vastinelementin välinen tekstipätkä sisältää toisia elementtejä, kuten kuvassa 1. Sisäkkäisyyden määrällä ei ole rajoitusta [B+00], mutta jokaisella elementillä on oltava vastinelementti eivätkä vastinelementit saa olla lomittain.

```
<henkilo>
    <nimi>
        <etunimi>John</etunimi>
        <sukunimi etuliite="von">Neumann</sukunimi>
    </nimi>
    <syntymaaika>
        <paiva>28</paiva>
        <kuukausi>12</kuukausi>
        <vuosi>1903</vuosi>
    </syntymaaika>
</henkilo>
```

Kuva 1: Elementtien sisäkkäisyys

Dokumentin hierarkkinen rakenne muodostuu sisäkkäisistä elementeistä. Kuvassa 1 esitetyllä esimerkillä on hierarkkinen rakenne, joka on esitetty puukuviona kuvassa 2. Puun juuri on elementti `<henkilo>` ja sen lehtinä ovat elementit `<etunimi>` sekä `<sukunimi>`, `<paiva>`, `<kuukausi>` ja `<vuosi>`. XML-dokumentissa on aina oltava

juurielementti eli *dokumenttielementti*, jonka sisällä kaikki muut elementit ja tekstit ovat [B+00]. Tätä esimerkkiä käytetään jatkossa sekä rakennemäärittelyä että esityksen tuottamista käsittelevissä luvuissa.



Kuva 2: Esimerkki XML-dokumentin puurakenteesta.

Elementeillä voi olla niitä kuvailevia *attribuutteja*. Attribuutille asetetaan arvo yhtäsuuruus- ja lainausmerkkejä käyttäen. Kuvan 1 esimerkissä elementillä `<sukunimi>` on attribuutti `etuliite`.

2.2 XML-dokumentin rakennemäärittely

Dokumentin tyyppi määritetään rakennemäärittelyssä, *DTD:ssä* (Document Type Definition) [B+00]. Rakennemäärittelyssä määritetään täsmällisesti, mitä eri elementit saavat sisältää ja missä järjestyksessä. Rakennemäärittely voi olla XML-dokumentin alussa, mutta yleensä se on omassa tiedostossaan.

```
<!ELEMENT henkilo (nimi, syntymaaika?)>
<!ELEMENT nimi (etunimi?, sukunimi)>
<!ELEMENT syntymaaika (paiva?, kuukausi?, vuosi)>
<!ELEMENT etunimi (#PCDATA)>
<!ELEMENT sukunimi (#PCDATA)>
<!ELEMENT paiva (#PCDATA)>
<!ELEMENT kuukausi (#PCDATA)>
<!ELEMENT vuosi (#PCDATA)>

<!ATTLIST sukunimi
          etuliite CDATA #IMPLIED>
```

Kuva 3: Kuvan 1 esimerkin rakennemäärittely.

Kuten kuvan 3 esimerkistä havaitaan, elementtien määritys alkaa rakennemäärittelyssä merkinnällä `<!ELEMENT` ja päättyy kulmasulkeeseen `>`. Aloitusta seuraa elementin nimi ja kaarisulkeissa annetaan elementin sisältö. Sisältö voi olla toisia elementtejä ja tekstiä, jonka merkintä on `#PCDATA`. Elementin sisällön osat voidaan luetella järjestyksessä pilkulla erottaen, jolloin järjestys on sitova. Käyttämällä pystyviivaa

pilkun sijasta voidaan määritellä vaihtoehtoiset sisällön osat. Välittömästi kunkin sisällön osan perään voidaan lisätä vaatimus osan peräkkäisten esiintymiskertojen rajoituksista. Rajoitusta osoittavat merkit ovat [B+00]:

Jos merkkiä ei ole, osa esiintyy tasan kerran.

? Osa voi esiintyä kerran tai olla esiintymättä lainkaan.

***** Osan peräkkäisten esiintymiskertojen määrä on rajoittamaton. Osa voi olla myös esiintymättä.

+ Osan peräkkäisten esiintymiskertojen enimmäismäärä on rajoittamaton mutta osa esiintyy ainakin kerran.

Sulkeita käyttämällä saadaan aikaan monimutkaisiakin sääntöjä siitä, mitä elementit voivat sisältää. Sääntöjen tiukkuus on rakennemäärittelyn suunnittelijan päätettävissä. Kuvan 3 esimerkkirakennemäärittely on varsin väljä, sillä se sallii henkilön etunimen jättämisen pois.

Elementin attribuuttien määritys rakennemäärittelyssä alkaa merkinnällä `<!ATTLIST` ja päättyy kulmasulkeeseen `>`. Merkintää seuraa elementin nimi. Nimen jälkeen määritetään listana attribuutit. Listassa jokainen attribuutti määritetään muodossa `nimi tyyppi vaadittavuus` [B+00].

Yleisimpiä XML-pohjaisissa opetuskielissä käytettäviä attribuutin tyyppejä ovat [B+00]:

CDATA Merkkijono.

NMTOKEN Useampi merkkijono välilyönnillä erotettuna.

ID Tunniste, joka saa esiintyä vain kerran XML- dokumentissa. Samalla elementillä voi olla vain yksi

IDREF Viittaus johonkin tunnisteeseen. Viitatun tunnisteen on esiinnyttävä XML-dokumentissa.

IDREFS Viittaus useampaan tunnisteeseen. Viitatuiden tunnisteiden on esiinnyttävä XML-dokumentissa.

Attribuutin vaadittavuuden vaihtoehdot ovat:

#IMPLIED Attribuutti voidaan XML-dokumentissa jättää elementistä pois eikä sillä ole oletusarvoa.

#REQUIRED Attribuuttia ei voi jättää elementistä pois.

#FIXED Attribuutilla on oletusarvo, joka annetaan merkkijonona lainausmerkkien sisällä tämän sanan jälkeen. XML-dokumentissa attribuutti voidaan jättää merkitsemättä, mutta sille ei voi asettaa muuta arvoa.

Rakennemäärittelyssä voidaan määritellä myös *entiteettejä*. Näiden avulla on mahdollista määrittää lyhenne pitkälle merkkijonolle ja käyttää lyhennettä pitkän merkkijonon sijaan rakennemäärittelyssä tai XML-dokumentissa. Entiteettien avulla on myös mahdollista sisällyttää rakennemäärittelyjä ja XML-dokumentteja toisiinsa. Entiteetin määrittely alkaa rakennemäärittelyssä merkinnällä `<!ENTITY %` , jota seuraa entiteetin nimi ja lainausmerkeissä entiteetin sisältämä teksti. Jos entiteetti on toinen rakennemäärittely, lainausmerkeissä on haettavan tiedoston URI-osoite ja lainausmerkkejä edeltää joko määre `SYSTEM` tai `PUBLIC`. Määritys päättyy kulmasulkeeseen `>`.

2.3 XML-dokumentin hyvinmuodostuneisuus, validius ja ilmentymät

XML-dokumentilla on kaksi oikeellisuustasoa. Dokumentti on *hyvinmuodostettu*, jos merkkaus noudattaa kaikilta osin XML-kielen määritystä [B+00]. Hyvinmuodostetussa XML-dokumentissa on alussa käytettävän XML-version määritys ja loppudokumentilla on juurellinen puurakenne. Jos lisäksi XML-dokumentilla on rakennemäärittely, jota XML-dokumentti täsmällisesti noudattaa, XML-dokumentti on *validi*.

XML-dokumentti on rakennemäärittelynsä mukaisen dokumenttityypin *ilmentymä*. Aiemmin esimerkkinä esitetyssä henkilödokumentissa voisi olla kenen tahansa henkilön tiedot. Tuo esimerkki on eräs henkilödokumenttityypin ilmentymä, jossa henkilönä on John von Neumann. On tärkeää huomata, että XML-merkkausta käytetään nimenomaan rakenteisten dokumenttien merkkaamiseen ja syntyvä rakennemäärittely sallii useita ilmentymiä. Useiden ilmentymien merkitys korostuu XML-dokumenttien jatkokäsittelyssä.

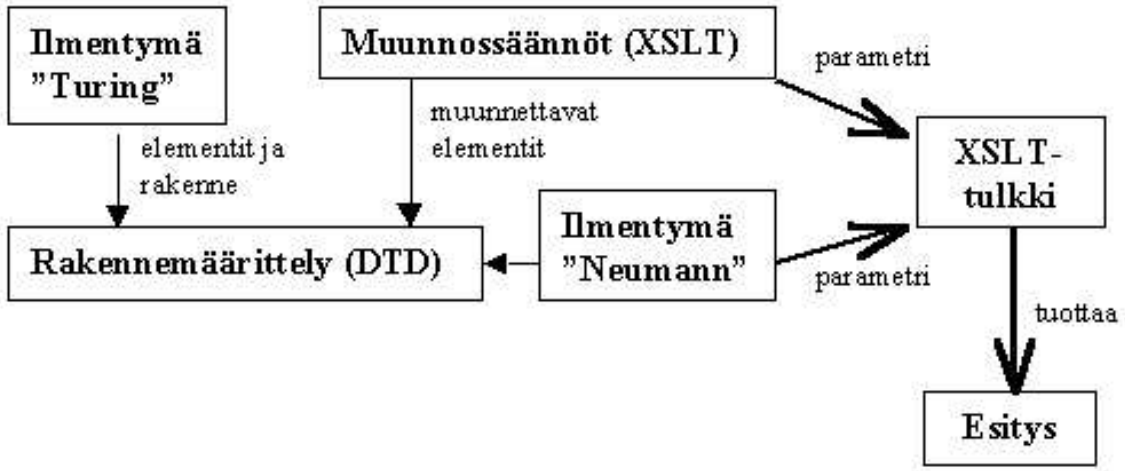
2.4 Esityksen tuottaminen

Dokumenttityypin ilmentymien esitys tuotetaan XSL-kielellä (The Extensible Stylesheet Language) [W3C02]. XSL-kieleen kuuluu kolme osaa: XML-dokumenttien osiin viittaamiseen luotu XPath (XML Path Language), XML-dokumenttien muuntamiseen kehitetty XSLT (XSL Transformations) ja täsmällisen ulkoasun ilmaisemiseen kehitetty XSL-FO (XSL Formatting Objects).

XSLT-kielen avulla voidaan dokumenttityypin ilmentymät muuntaa toiseen muotoon eli tuottaa niille esitys. XSLT-muunnostiedostossa määritetään jokaiselle merkkauksessa käytetylle elementille toimenpide, jonka muunnoksen tekevä ohjelma, *XSLT-tulkki*, suorittaa kohdatessaan kyseisen elementin XML-dokumentissa. Toimenpiteiden tulokset kirjoitetaan erilliseen tiedostoon. Kun muunnos tehdään HTML:ksi, toimenpide on yleensä HTML:n elementtien kirjoittamista kohdatun XML-dokumentin elementin merkkaaman tekstin ympärille.

Esityksen tuottamisen pääpiirteet on esitetty kuvassa 4. Kuvassa on rakennemäärittely, jota noudattavat sekä John von Neumannin että Alan Turingin tiedot sisältävät ilmentymät. Muunnossäännöt eli XSLT-tiedosto on tehty rakennemäärittelyn määrittelemälle dokumenttityypille, joten säännöt pätevät kaikille rakennemäärittelyä noudattaville ilmentymille. Kuvassa ilmentymistä on valittu John von Neumannin tiedot sisältävä ilmentymä, joka yhdessä muunnossääntöjen kanssa annetaan parametreina XSLT-tulkille. XSLT-tulkki suorittaa muunnossäännöt ja tuottaa ilmentymälle HTML-esityksen. Jos XSLT-tulkille annettaisiin parametrina John von Neumannin tiedot sisältävän ilmentymän sijasta Alan Turingin tiedot sisältävä ilmentymä, tuloksena tulkki tuottaisi tämän ilmentymän HTML-esityksen.

Esityksen tuottamisessa käytettävät muunnossäännöt kirjoitetaan tiedostoon. Kuvassa 5 on esimerkki tällaisesta muunnossäännöt sisältävästä XSLT-tiedostosta, jonka avulla henkilöesimerkin rakennemäärittelyä noudattavat ilmentymät voidaan muuttaa HTML-muotoon. Esimerkistä nähdään, että elementissä `<xsl:template match=''henkilo''>` attribuutin `match` arvo määrittää, mille käsiteltävän ilmentymän elementille muunnossääntöjä sovelletaan. Tämän jälkeen kirjoitetaan normaalit HTML-tiedoston otsikkomerkinnät. Elementti `<xsl:apply-templates />` on XSLT-tulkille käsky edetä ilmentymän käsittelyssä käsiteltävänä olevan elementin lapsielementteihin ja soveltaa niille määritettyjä muunnossääntöjä. Koska käsittely on rekursiivista, palataan lopuksi rekursiosta takaisin ja lisätään HTML-tiedoston lopussa tarvittavat vastinelementit `</body>` ja `</html>`. Ilmentymän elementeille `<nimi>` ja `<syntymaaika>` määritetyt muunnossäännöt on tehty samoin. Näissä kuitenkin



Kuva 4: Esityksen tuottaminen ilmentymille.

elementtien sisältämien elementtien merkkaamat tekstit on suoraan poimittu määrittämällä `select`-attribuutilla käsiteltävät elementit. Lisäksi on käytetty `text()`-funktiota, joka palauttaa ilmentymässä elementtien merkkaaman tekstin. John von Neumannin tiedot sisältävälle ilmentymälle on tämän XSLT-tiedoston avulla tuotettu XSLT-tulkilla HTML-esitys, joka esitetään kuvassa 6.

XSL-kielessä voidaan käyttää hyväksi myös toistolauseita, muuttujia ja joitakin valmiiden funktioiden palauttamia arvoja. Tämän seurauksena muunnoksen tuloksena syntyneen esityksen rakenne ei ole sidottu alkuperäisen XML-dokumenttityypin rakenteeseen. Lisäksi esitykseen saadaan helposti mukaan juuri käsiteltävään XML-dokumenttityypin ilmentymään liittyvää tietoa, esimerkiksi otsikoiden numerointi. XSLT-tiedosto on itsekin XML-dokumentti, jonka on vastattava kielen kehittäjän laatimaa XSLT-dokumentin rakennemäärittelyä [C+99].

XSLT-muunnoksen tuloksena voidaan määrittää syntymään myös XSL-FO-kieltä. Tällöin XSLT-tiedostossa määritettävissä toimenpiteissä kirjoitetaan HTML:n sijasta XSL-FO-käskyjä [W3C02]. Tuloksena syntyvä fo-tiedosto on tarkka kuvaus käsitellyn XML-dokumenttityypin ilmentymän ulkonäöstä. Esitys voidaan muuntaa XSL-FO-muodosta useisiin käytössä oleviin sähköisiin dokumenttiformaatteihin [Apa03a].

XML- ja XSLT-tekniikoiden käyttö mahdollistaa samanrakenteisten dokumenttien tehokkaan tuottamisen. Samaa rakennemääritystä noudattavat dokumenttityypin ilmentymät voidaan kääntää samaa XSLT-tiedostoa käyttäen. Näin ilmentymien esityksistä saadaan automaattisesti yhdenmukaisen näköiset. Etenkin ohjelmisto-

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" indent="yes" encoding="iso-8859-1"/>

  <xsl:template match="henkilo">
    <html>
      <head><title>Henkilö</title></head>
      <body>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="nimi">
    <h1>
      <xsl:apply-templates select="etunimi/text()" /> <xsl:text> </xsl:text>
      <xsl:apply-templates select="sukunimi/text()" />
    </h1>
  </xsl:template>

  <xsl:template match="syntymaaika">
    <p>Tämä henkilö syntyi
      <xsl:apply-templates select="paiva/text()" />.
      <xsl:apply-templates select="kuukausi/text()" />. vuonna
      <xsl:apply-templates select="vuosi/text()" />.
    </p>
  </xsl:template>
</xsl:stylesheet>
```

Kuva 5: Esimerkki XSLT-muunnostiedostosta.

```
<html>
<head>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>
  Henkil&ouml;
  </title>
</head>
<body>
  <h1>John Neumann</h1>
  <p>
    T&auml;m&auml; henkil&ouml; syntyi
    28.
    12.
    vuonna
    1903.
  </p>
</body>
</html>
```

Kuva 6: XSLT-tiedoston avulla tuotettu HTML-esitys.

tuotannossa onkin perusteltua käyttää samanrakenteisten käyttöohjeiden ja muiden dokumenttien tuottamiseen XML-tekniikoita, koska dokumenttien rakenteet ovat yleensä hyvin samanlaisia ja XML- ja XSLT-tiedostojen kirjoittaminen voidaan antaa eri työntekijöiden tehtäväksi [Dea02].

XML-tekniikoiden tehokkuus ja monipuolisuus ovat herättäneet kiinnostusta myös opetusmateriaalin tuottajien piirissä. Koska XML- tulkkeja on yleisesti saatavilla [Apa03a, Apa03b], jopa yksittäiset opettajat voivat käyttää XML-tekniikoita materiaaliensa tuottamiseen.

3 XML-metakielen käyttötavat oppimateriaalien tuottamisessa

XML-metakielen käytössä oppimateriaalien tuotannossa on havaittavissa kaksi pääsuuntausta. Metadatan liittäminen valmiisiin oppimateriaaleihin on useiden järjestöjen työalueena ja oppimateriaalien tuottaminen kokonaan XML-metakielisinä on erityisesti joidenkin yliopistojen pyrkimyksenä.

3.1 Metadatan liittäminen valmiisiin oppimateriaaleihin

Liittämällä oppimateriaaliin metadataa pyritään oppimateriaalin parempaan löydettävyyteen, uudelleenkäytettävyyteen ja toimintaan eri järjestelmissä. Tämän vuoksi monet organisaatiot ovat kehittämässä standardeja siitä, mitä metatietoa ja millaisessa muodossa oppimateriaaleihin tulisi liittää. Eri organisaatiot käyttävät toistensa kehittämiä ehdotuksia omia standardiluonnoksia luodessaan ja tavoitteena on lopulta saada aikaan yhteinen ISO:n hyväksymä standardi, joka sisältäisi metadatan liittämisen lisäksi myös muita verkko-oppimisympäristöjen oppimateriaalin tuottamisessa käytettäviä määrityksiä [Son02].

Pohjana XML-kieleen perustuville opetusmateriaalin standardiluonnoksille toimii IEEE:n vuonna 2002 hyväksymä LOM-standardi (Learning Object Metadata) [H$^+$02]. Tässä standardissa määritetään täsmällisesti, mitä metatietoa *oppimisaihioihin* (Learning Objects) tulee liittää. Oppimisaihiolla tarkoitetaan mitä tahansa opetuksessa käytettävää materiaalia, esimerkiksi luentomonistetta, HTML-sivua tai videoleikettä. Standardissa oppimisaihioon liitettävä metatieto on jaettu yhdeksään pääaiheeseen, jotka ovat: yleinen tieto, kehityshistoria, liitettävän metadatan kuvaus, tekniset

vaatimukset ja ominaisuudet, pedagogiset ja koulutukselliset ominaisuudet, tekijänoikeudet, suhteet muihin oppimisaihioihin, oppimisaihion käyttöön liittyvät kommentit sekä oppimisaihion luokitus. Nämä pääaiheet on jaettu pienemmiksi osiksi, joissa määritetään hyvinkin tarkasti, mistä tiedoista aihealue koostuu. Tämä hierarkkinen rakenne sekä rakenteen sisältämien tietojen tyypit on tiukasti määritetty. Standardi ei kuitenkaan määrää metakieltä, jolla metatiedon kuvaaminen toteutetaan [H+02].

Oppimateriaaliin liitettävän metadatan kuvaus kirjoitetaan yleensä erilliseen tiedostoon. Metadata on tiedostossa kirjoitettu metakielellä ja metatiedolla on rakenne. Tästä kuvauksesta käytetään joissain yhteyksissä myös nimeä tietomalli [MT01]. LOM-standardi määrittää tietomallin rakenteen. Kun LOM-standardin toteutuksessa käytetään XML-metakieltä, tämä rakenne annetaan XML-dokumentin rakennemäärittelynä.

IMS (IMS Global Learning Consortium, alunperin Instructional Management Systems) kehittää omaa määritystään metadatan liittämisestä oppimateriaaliin. Määritys koostuu yleisen tason tietomallin määrityksestä sekä sen XML-toteutuksen määrityksestä. Tietomallin määrityksen versiot perustuvat LOM-standardin kehitysaikaisiin versioihin, joihin on tehty vain pieniä muutoksia [MT01, H+02]. Tämän määrityksen lisäksi IMS kehittelee myös muita määrityksiä koskien oppimateriaalin sisällön pakkausta, pedagogiikan lisäämistä verkko-oppimismateriaaleihin, kokeiden laatimista sekä oppijakohtaisen tiedon keräämistä. IMS tekee läheistä yhteistyötä LOM-standardin kehittäneen IEEE Learning Standard Committee -ryhmän sekä Authoring and Distribution Networks for Europe (ARIADNE) -järjestön kanssa. IMS:n kanssa yhteistyötä tekee myös The Aviation Industry CBT Committee (AICC), joka kehittää oppimateriaalin hallintajärjestelmien välisiä standardeja [Son02].

Kuvassa 3.1 on esimerkki siitä, miten IMS:n LOM-standardille määrittelemän XML-sovelluskielen mukaisesti oppimisaihioita voidaan kuvata. Esimerkissä on kuvattu yleiset tiedot John von Neumannin henkilötiedot sisältävästä HTML-sivusta. Aluksi on annettu oppimisaihion otsikko, oppimisaihion sijainti sekä kirja, jossa oppimisaihio esiintyy. Lisäksi on annettu tunnus sekä kuvattu oppimisaihiota suomeksi ja englanniksi. On huomattavaa, että tämä esimerkki kuvaa ainoastaan LOM-standardin yleisten tietojen osion. Kokonainen kuvaus, joka sisältäisi myös muiden LOM-standardin osioiden mukaista metatietoa, olisi huomattavasti pidempi ja työläämpi kirjoittaa.

USA:n työministeriön alainen ADL (The Advanced Distributed Learning) kehit-

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE lom SYSTEM "imsmd_rootv1p2.dtd">
<lom>
    <general>
        <title>
        <langstring xml:lang="x-none">John von Neumann</langstring>
    </title>
    <catalogentry>
       <catalog>URI</catalog>
       <entry>
       <langstring xml:lang="x-none">john_von_neumann.html</langstring>
       </entry>
    </catalogentry>
    <catalogentry>
       <catalog>Tietojenkäsittelytieteen historia</catalog>
       <entry>
       <langstring xml:lang="fi">140403</langstring>
       </entry>
    </catalogentry>
        <description>
        <langstring xml:lang="fi">
                Tämä seminaariaine antaa kuvan John von
                Neumannin elämästä ja hänen keksinnöistään.
            </langstring>
        <langstring xml:lang="en">
                This essay is about the life of John von Neumann and his
                inventions.
            </langstring>
    </description>
    </general>
</lom>
```

Kuva 7: Esimerkki IMS:n mukaisesta metadatasta.

tää SCORM-määritystä (Sharable Content Object Reference Model), joka käyttää hyväkseen IMS:n määrityksiä sekä LOM-standardia. SCORM on yritys kerätä yhteen eri tahoilla kehitellyt www-pohjaisen oppimateriaalin tuottamiseen liittyvät tekniset määritykset. Siksi metatiedon liittäminen oppimateriaaliin on vain pieni osa SCORM-määritystä [Adv02].

3.2 Oppimateriaalin tuottaminen XML-metakielellä

Opetusdokumentit voidaan tuottaa käyttämällä XML-metakieltä. Aiemmin on jo esitetty, miten yksittäiset dokumentit voidaan merkata ja merkkaus voidaan määritellä dokumentin rakennemäärityksessä. Kirjoittamalla XSLT-muunnostiedosto voidaan samaa rakennemäärittelyä noudattaville dokumenttityypin ilmentymille tuottaa XSLT-tulkilla esitys. Jos samanrakenteisia dokumentteja käytetään opetuksessa paljon, voi tämä tekniikka olla tehokas tapa vähentää dokumenttien taittoon tarvittavaa työtä.

Tarve oppimateriaalin monikäyttöisyyteen on laajentanut XML-metakielen käytön kuvaamaan myös kokonaisia kursseja. Etenkin verkko-oppimisympäristöissä voidaan käyttää hyväksi oppijan toimintaan mukautuvaa eli *adaptiivista* oppimateriaalia [SFB99]. Tämä kuitenkin edellyttää, että opetettavien asioiden järjestys ja esitys eivät riipu pelkästään tietystä opetusdokumentista. Ongelmaa voidaan lähestyä rakentamalla opetettavan kurssin aihealueen asioista *käsitemalli*, joka kuvaa asiat ja asioiden väliset suhteet. Näin opetusmateriaalin käyttömahdollisuudet laajenevat, kun samasta materiaalista voidaan tuottaa erilaisia esityksiä oppilaiden esitietojen mukaan ja samaa materiaalia voidaan käyttää sekä verkko-oppimisjärjestelmissä että paperiversioina luokkaopetuksessa [Süß00a].

Kokonaisten kurssien ja opetusaihealueiden kuvaaminen ei ole yhtä laajalti tutkimuksen kohteena kuin metadatan liittäminen jo olemassa oleviin opetusdokumentteihin. Metadatan liittämistä ovat kehittämässä useat kansainväliset järjestöt, mutta opetusmateriaalin tuottamista XML-metakielen avulla kehittävät ennemminkin yksittäiset yliopistot omiin tarpeisiinsa. Konferenssijulkaisuja tarkasteltaessa esiin nousevat saksalaiset Passaun yliopiston ja Dresdenin teknillisen yliopiston tutkimusryhmät [Süß00a, WL01]. Ne kehittävät omia XML-metakieltä käyttäviä oppimateriaalin hallintajärjestelmiään, joista on jo olemassa toimivia versioita. Passaun yliopiston tutkimusryhmän kehittämä opetusmateriaalin kuvaamiseen tarkoitettu XML-sovelluskieli on LMML [SFB99]. Dresdenin teknillisen yliopiston kehittämä kieli on TeachML [WL01].

Jako metadatan lisäämiseen valmiiseen oppimateriaaliin ja oppimateriaalin tuottamiseen XML-metakielellä ei ole täysin selvä. Eri yliopistojen luomia XML-sovelluskieliä on kehitetty ottamaan paremmin huomioon niiden kehitysvaiheessa täsmentynyt LOM-standardi sekä IMS-määritykset. Näissä sovelluskielissä esiintyy LOM-standardin mukaisia elementtejä ja metatiedon rakennetta. Lisäksi voi olla mahdollista liittää XML-sovelluskielellä tehtyihin oppimateriaalin osiin IMS-määritysten mukaiset kuvaustiedostot [Süß00b, WL01]. Monissa XML-sovelluskieltä käyttävissä oppimateriaalin hallintajärjestelmissä on mahdollista liittää myös valmiita rakenteettomia dokumentteja osaksi kurssin tai opetusaihealueen kuvausta [SFB99, WL01]. Tässä tutkielmassa on kuitenkin pidetty jaon perustana sitä, että lisäominaisuuksista huolimatta tällaisissa järjestelmissä pääpaino on edelleen oppimateriaalin tuottamisessa XML-metakielellä.

3.3 Edut ja haitat

Metadatan liittäminen oppimateriaaliin tarkentaa hakutuloksia materiaalia etsittäessä, koska hakuja voidaan kohdistaa myös materiaalin semanttisiin ominaisuuksiin. Lisäksi hakuja voidaan suorittaa oppimateriaalin hallintajärjestelmissä ja verkko-oppimisalustoissa, jotka käyttävät samaa metadatan esityksen määritystä [MT01].

Järjestelmissä, joissa opetusdokumentit ovat XML-muodossa, tiedon löytyminen on helppoa. Tällöin opetusdokumentit sisältävät metatietoa, jonka perusteella hakuja voidaan suorittaa [SFB99]. Tiedon löytyminen rajoittuu kuitenkin oppimateriaalin hallintajärjestelmän sisälle, mikäli hallintajärjestelmä ei ulospäin tarjoa yleisen määrityksen mukaista metatietoa materiaalista [HSF01, SFB99]. Tämän vuoksi joissakin oppimateriaalin kuvauskielissä on mahdollista tuottaa materiaalin metatiedosta myös IMS:n määrityksen mukainen esitys [WL01] ja kuvauskielissä käytettävät attribuutit voivat olla samoja kuin esimerkiksi LOM-standardissa [Süß00b].

XML-metakielen käytöllä pyritään alustariippumattomuuteen. Kun metadata liitetään valmiisiin opetusdokumentteihin, opetusdokumenttien lukemiseen tarvittava ohjelma voidaan lukea metadatasta [H$^+$02]. Dokumentin lukeminen ei kuitenkaan onnistu, ellei ohjelmaa ole saatavissa. Opetusdokumentteja voidaan hakea toteutuksiltaan erilaisista opetusmateriaalin hallintajärjestelmistä, mutta jokaiselle mahdolliselle dokumenttiformaatille on oltava jokaisella hallintajärjestelmällä lukuohjelmat. Tämän ongelman ratkaiseminen on ollut lähtökohta niissä järjestelmissä, joissa oppimateriaali on kuvattu XML-muodossa. Tällöin XML-tiedostojen esitys voidaan tuottaa käytössä olevassa dokumenttiformaatissa [SFB99]. Lisäksi XML-työkaluista

on saatavissa Java-pohjaisia versioita, jotka voivat toimia eri käyttöjärjestelmissä ja jotka ovat maksuttomia [Apa03b, Apa03a].

4 LMML-opetuskieli ja sen käsitemalli

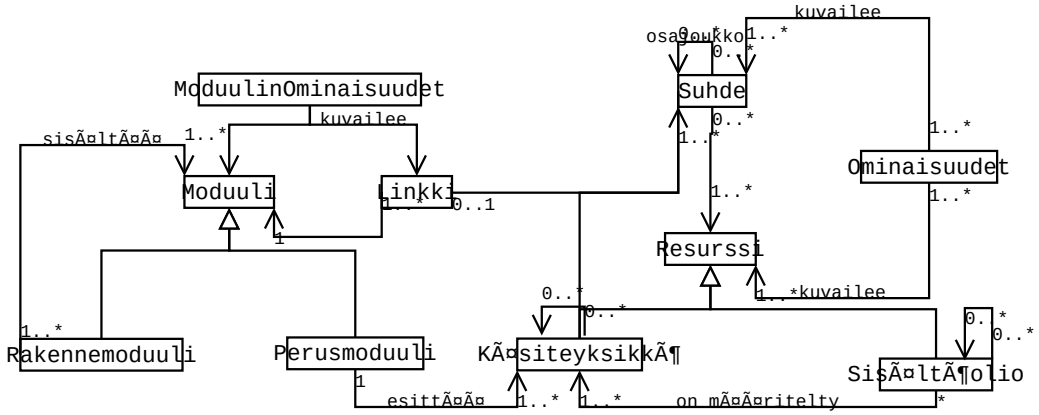
Esimerkkinä XML-pohjaisesta opetuskielestä, jolla mallinnetaan koko opetusaihealuetta, on saksalaisen tutkimusryhmän kehittämä LMML (Learning Material Markup Language). Opetusaihealueen mallinnuksesta muodostuu käsitemalli, joka toteutetaan XML-sovelluskielen LMML avulla.

4.1 Käsitemalli

Käsitemallissa, jonka LMML-kieli [SFB99] toteuttaa, kuvataan eri aihealueisiin liittyvien dokumenttien käsitteellistä rakennetta sekä dokumenttien välisiä ja materiaalin selaamiseen eli navigointiin vaikuttavia suhteita. Näitä voivat olla esimerkiksi dokumentin vaikeustaso ja sen lukemisessa tarvittavat esitiedot. Tätä käsitemallia kutsutaan pohjamalliksi ja se on esitetty kuvassa 8. Pohjamallin rakenne on suoraviivaisesti johdettu havainnosta, että eri aihealueisiin liittyvän materiaalin voidaan ajatella semanttisesti koostuvan erilaisista *käsiteyksiköistä* (conceptual unit), joilla on erilaisia *sisältöolioita* (content object) [Süß00a]. Sisältöolio voi olla esimerkiksi rakenteetonta tekstiä tai toinen sisältöolio ja näillä kaikilla voi olla suhteita toisiinsa.

Kuvassa 8 esitetyssä pohjamallissa on nähtävissä pienempiä kokonaisuuksia: `Moduuli` ja siitä periytetyt `Rakennemoduuli` sekä `Perusmoduuli` antavat mahdollisuuden koota yhteen eri käsitteitä. Rakennemoduulissa voikin olla esimerkiksi tieto kaikista tiettyyn kurssiin kuuluvista käsiteyksiköistä. Linkkien avulla voidaan esittää moduulien, käsiteyksiköiden ja sisältöolioiden välisiä suhteita, jotka vaikuttavat navigointiin. Moduuleihin, linkkeihin, suhteisiin, käsiteyksiköihin sekä sisältöolioihin voidaan liittää kuvailevaa tietoa luokkien `Ominaisuudet` ja `ModuuliOminaisuudet` avulla.

Materiaalien eri aihealueisiin erikoistuneet mallit perustuvat edellä esiteltyyn pohjamalliin. Jokaisella aihealueella on omanlaisensa sisältöyksiköt, sisältöoliot ja näiden väliset suhteet. Eri aihepiireihin erikoistuneiden mallien kuvauksissa pohjamallin käsitteille on annettu aihepiireihin liittyvät konkreettiset vastineet ja kuvauksien suhteet vastaavat pohjamallin abstraktien käsitteiden suhteita [Süß00a]. Kuvassa 8 esitettyä pohjamallia voitaisiin erikoistaa vastaamaan tietojenkäsittelytieteen opetusmateriaalin rakenteita. Tällöin sisältöolioita voisivat olla esimerkiksi `todistus`



Kuva 8: LMML-kielen pohjamalli. [Süß00a]

ja `algoritmi`.

4.2 XML-toteutus

Erikoistuneet mallit kuvataan XML-metakielellä. Jokainen sisältöolio ja käsiteyksikkö voidaan kuvata omassa XML-tiedostossa (kuvat 9, 10 ja 11). XML:n ominaisuus liittää XML-tiedostoja osaksi toista XML-tiedostoa antaa mahdollisuuden saada aikaan modulaarinen rakenne, jossa sisältöolioita ja käsiteyksiköitä voidaan käyttää eri XML-dokumenttien osina [B+00].

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE lmml SYSTEM "LMML-CS.dtd">
<lmml version="-//DE.UNI-PASSAU.DAISY//DTD LMML-CS 1.1//EN">
  <collection language="fi" author="Ahti Syreeni" label="binääripuu" title="Binääripuu">
    <definition uri="binaaripuu_maaritelma.xml" />
    <example uri="binaaripuu_esimerkki.xml" />
  </collection>
</lmml>
```

Kuva 9: Esimerkki LMML-CS -kielellä kuvatusta käsiteyksiköstä binääripuu, joka koostuu määritelmästä sekä yhdestä esimerkistä.

LMML-opetuskielen rakennemäärittely koostuu LMML-kehyksestä sekä aihealuekohtaisesta laajennoksesta [Süß00b]. LMML-kehykseen liittyvät tiedostot määrittelevät kaikille LMML-kielen aihealuekohtaisille sovelluksille yhteisiä elementtejä, attribuutteja ja entiteettejä. Esimerkkejä tällaisista ovat elementit `<definition>` ja `<example>`, joita on käytetty kuvissa 9, 10 ja 11.

Aihealuekohtaisissa laajennoksissa määritetään aihealueen kuvaamisessa tarvitta-

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE lmml SYSTEM "LMML-CS.dtd">
<lmml version="-//DE.UNI-PASSAU.DAISY//DTD LMML-CS 1.1//EN">
  <definition title="binääripuu" language="fi" difficulty="low">
    <LMMLtext>
      Binääripuu on 2-haarainen puu, jossa jokaiseen solmuun liittyy
      täsmälleen kaksi alipuuta.
    </LMMLtext>
  </definition>
</lmml>
```

Kuva 10: LMML-CS -kielellä kuvattu sisältöolio. Sisältöolio on tässä binääripuun määritelmä.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE lmml SYSTEM "LMML-CS.dtd">
<lmml version="-//DE.UNI-PASSAU.DAISY//DTD LMML-CS 1.1//EN">
  <example title="yksinkerainen binääripuu" language="fi" difficulty="low">
    <LMMLtext>Puu, jonka juurella on kaksi lehteä on, on binääripuu.</LMMLtext>
  </example>
</lmml>
```

Kuva 11: LMML-CS -kielellä kuvattu sisältöolio. Sisältöolio on tässä esimerkki binääripuusta.

via LMML-kehyksen sisältämättömiä elementtejä, attribuutteja ja entiteettejä. Esimerkkejä näistä ovat tietojenkäsittelytieteeseen tehdyn laajennoksen elementit `<algorithm>` ja `<proof>` [Süß00a].

LMML on kieliperhe, joka koostuu aihealuekohtaisista LMML-kielen sovelluskielistä. Sovelluskieli syntyy, kun LMML-kielen rakennemäärityst laajennetaan uusilla elementeillä, attribuuteilla ja entiteeteillä. Edellä esitetty tietojenkäsittelytieteeseen tehty laajennos on LMML-kielen sovelluskieli, jonka nimeksi on annettu LMML-CS. Sovelluskieliä on monia, koska LMML:n perusidea on, että uusia aihealueita varten on tarkoitus tehdä oma laajennos [SFB99].

4.3 Edut ja haitat

LMML:n käsitemallin mukaan tuotetussa oppimateriaalissa tieto on helposti löydettävissä. Käsitemallista voidaan hakea sisältöolioita niiden tyypin mukaan ja näin

oppimateriaalista voidaan esimerkiksi hakea kaikki esimerkit tai määritelmät. Samoin sisältöolioiden suhteiden mukaan hakeminen onnistuu, joten on mahdollista hakea esimerkiksi kaikki tietyn todistuksen oppimateriaalissa esiintyvät perustelut [SFB99]. Nämä ominaisuudet mahdollistavat adaptiivisen navigoinnin, jossa asioiden käsittelyjärjestys perustuu oppijan esitietoihin ja tekemiin valintoihin. Esimerkiksi oppijalle voidaan esittää oppimateriaalista hänen esitietoihinsa nähden sopivan vaikeat määritelmät ja esimerkit.

Käsitemallissa on mahdollista lisätä sisältöolioita ja käsiteyksiköitä. Oppimateriaalia voidaan lisätä luomalla uusia LMML-dokumentteja. Kuten binääripuuesimerkissä (kuvat 9, 10 ja 11) on esitetty, uuden materiaalin tekemisessä voidaan käyttää hyväksi jo olemassa olevia LMML-dokumentteja LMML-kielen modulaarisuuden vuoksi [Süß00a].

Uuden tyyppistä sisältöä voidaan määritellä aihealueisiin erikoistuneissa malleissa ja toteuttaa ne LMML-kielen sovelluskielissä [SFB99]. Tämä kuitenkin edellyttää uusien elementtien luomista LMML-sovelluskielessä ja näin ollen jokaisen uuden elementin lisääminen aiheuttaa tarpeen muuttaa sovelluskielen rakennemäärittelyä [WL01]. Opetusmateriaalin tuottajaa rajoittaa käytössä oleva rakennemäärittely ja rakennemäärittelyä muutettaessa uuden tyyppistä sisältöä sisältäviä dokumentteja ei voi käyttää vanhaa rakennemäärittelyä käyttävissä järjestelmissä.

5 Johtopäätökset

XML-metakielen avulla voidaan tuottaa tehokkaasti samanrakenteisia dokumentteja ja liittää niihin metatietoa. Tällöin sisältö voidaan erottaa tiedon esityksestä ja sisällölle voidaan luoda monia vaihtoehtoisia esitystapoja. Tämä mahdollisuus on tehnyt XML-tekniikoiden käytön oppimateriaalin tuottamisessa houkuttelevaksi, koska nykyisin oppimateriaalia käytetään monissa eri muodoissa, niin tietokoneella kuin perinteisesti paperillakin.

Tiedon esityksen ja sisällön erottamisen ideaa soveltamalla päädytään lopulta ajatukseen kokonaisten opetusaihealueiden ja kurssien kuvaamisesta. Yksittäiset opetusdokumentit sisältävät määrätyssä järjestyksessä tietoa opetusaihealueesta. Opetusdokumentit ovatkin itse asiassa opetusaihealueen tiedon esittämistä ja useat opetusdokumentit sisältävät samaa tietoa opetusaihealueesta. Tämän seurauksena on alettu tutkia mahdollisuutta kuvata kokonaisia opetusaihealueita modulaarisina tietokantamaisina rakenteina XML-metakielen avulla. Näin yksittäiset opetusdoku-

mentit voitaisiin tarpeen mukaan koostaa opetusaihealueen kuvauksen osista.

Oppimateriaalin hallintajärjestelmiä, joissa oppimateriaali tuotetaan XML-metakielellä, kehittävät lähinnä yliopistot omiin tarpeisiinsa. Pyrkimys opetusaihealueiden ja kurssien kuvaamiseen on näissä järjestelmissä hallitseva piirre. Järjestelmien yhteensopivuudessa on kuitenkin ongelmia, koska jokaisella järjestelmällä on oma XML-sovelluskielensä. Järjestelmien etuina ovat kuitenkin laitteistoriippumattomuus ja mahdollisuus tuottaa oppimateriaalista erilaisia ja eri formaateissa olevia esityksiä. Erityinen etu on, että tieto on hyvin modulaarista ja näin ollen tiedon esitys voidaan saada hyvinkin adaptiiviseksi.

Metadatan liittäminen valmiiseen oppimateriaaliin on muodostumassa hallitsevaksi suuntaukseksi XML-metakielen käytössä oppimateriaalituotannossa. Tällöin itse oppimateriaali on perinteisissä sähköisissä formaateissa, mutta jokaisesta dokumentista eli oppimisaihiosta on kirjoitettu kuvaus XML-metakielellä. Liitettävästä metatiedosta on olemassa vuonna 2002 hyväksytty LOM-standardi ja lisäksi monet kansainväliset organisaatiot kehittävät sen pohjalta laajempia sen XML-toteutukseen liittyviä standardiluonnoksia. Koska standardit ovat yleisesti hyväksyttyjä, yhteensopivuus näitä määrityksiä käyttävien oppimisalustojen välillä on hyvä. Oppimateriaalien katseluun tarvitaan kuitenkin alkuperäiset ohjelmat, joten laite- ja ohjelmistoriippumattomuudessa ei päästä samalle tasolle kuin järjestelmissä, joissa oppimateriaali tuotetaan XML-metakielellä. Metadatan liittäminen ei houkuttele opetusdokumentteja hienojakoisempaan modulaarisuuteen ja mahdollisuus adaptiivisuuden toteuttamiseen rajoittuukin usein navigointiin opetusdokumenttien välillä. Lisäksi tieto ei ole yhtä tarkasti käytettävissä uudelleen kuin opetusaihealueiden ja kurssien mallintamiseen perustuvissa järjestelmissä.

Lähteet

Adv02 Advanced Distibuted Learning, ADL SCORM Version 1.3 Application Profile, 2002. `http://www.adlnet.org/adldocs/Other/SCORMV1.3_SeqAppProfile.zip`. [17.2.2003]

Apa03a Apache Software Foundation, FOP, 2003. `http://xml.apache.org/fop/index.html`. [11.2.2003]

Apa03b Apache Software Foundation, Xalan-Java Overview, 2003. `http://xml.apache.org/xalan-j/overview.html`. [11.2.2003]

B+00 Bray, T. et al., Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, 2000. `http://www.w3.org/TR/2000/REC-xml-20001006`. [11.2.2003]

C+99 Clark, J. et al., XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999, 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`. [11.2.2003]

Dea02 Deach, S., What Is XSL-FO and When Should I Use It? *The Seybold Report*, 2,17(2002). [Myös `http://www.seyboldreports.com/TSR/free/0217/techwatch.html`] [11.2.2003].

H+02 Hodgins, W. et al., Draft Standard for Learning Object Metadata. Lopullinen LOM-standardi., IEEE, New York, 2002.

HSF01 Hollfelder, S., Stecher, R. ja Frankhauser, P., Tailoring Training Courses Using XML-based Metadata. *Proceedings of the Third DELOS Network of Excellence Workshop on Interoperability and Mediation in Heterogeneous Digital Libraries*, Darmstadt, Germany, September 2001. [Myös `http://www.ercim.org/publication/ws-proceedings/DelNoe03/5.pdf`] [11.2.2003].

MT01 McKell, M. ja Thropp, S., IMS Learning Resource Meta-Data Information Model (Version 1.2.1 Final Specification), 2001. `http://www.imsproject.org/metadata/imsmdv1p2p1/imsmd_infov1p2p1.html#11%69559`. [17.2.2003]

SFB99 Süß, C., Freitag, B. ja Brössler, P., Metamodeling for Web-Based Teachware Management. *Proc. ER'99 Conf. WWWCM'99 Workshop on the World-Wide Web and Conceptual Modeling*, Paris, France, November 1999. [Myös `http://daisy.fmi.uni-passau.de/publikationen/SFB99/paper.pdf`] [11.2.2003].

Son02 Sonwalkar, N., Demystifying Learning Technology Standards, Part I: Development and Evolution. *Syllabus*, 3,1(2002). [Myös `http://www.syllabus.com/article.asp?id=6134`] [11.2.2003].

Süß00a Süß, C., Adaptive Knowledge Management: A Meta-Modeling Approach and its Binding to XML. *12 Gl-Workshop Grundlagen Von Datenbanken*, Christian-Albrechts-Universit, Germany, 2000. [Myös `http://media.inhatc.ac.kr/papers/hypermedia/Sub00a.pdf`] [11.2.2003].

Süß00b Süß, C., LMML framework, LMML11-CS driver and modules archive, 2000. `http://daisy.fmi.uni-passau.de/pakmas/lmml/11/LMML11-CS-dtds.zip`. [11.2.2003]

W3C02 W3C, The Extensible Stylesheet Language (XSL), 2002. `http://www.w3.org/Style/XSL/`. [17.2.2003]

WL01 Wehner, F. ja Lorz, A., Developing Modular and Adaptable Courseware Using TeachML. *Proc. ED-MEDIA Conf. World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Tampere, Finland, Juni 2001. [Myös `http://www-mmt.inf.tu-dresden.de/english/projekte/Publikationen/details%/index.asp?p=0106.pdf`] [11.2.2003].