Luento 3: Tietorakenteiden esittäminen

AS-0.110 XML-kuvauskielten perusteet

Janne Kalliola

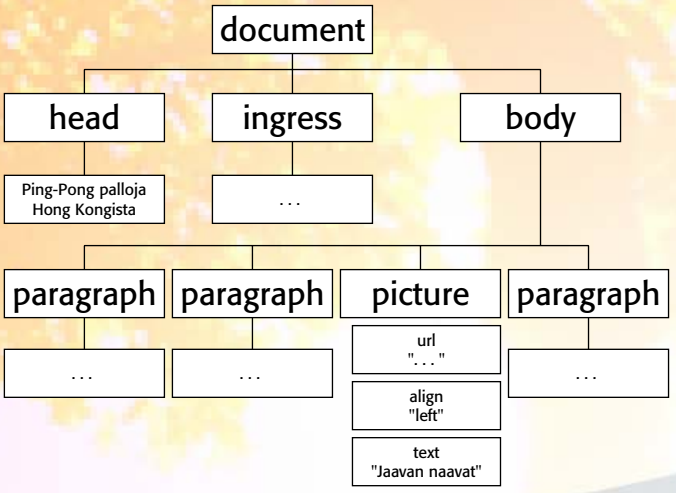
Tietorakenteiden esittäminen

- XML-dokumentti puuna
  - Muunnokset muodosta toiseen
- Perustietorakenteet
  - listat
  - puut
  - verkot
  - muut rakenteet
- Tietotyyppien kuvaaminen käytännössä
  - rakenteiden yhdistely
  - primitiivityypit

XML-dokumentti puuna

- XML-dokumentti voidaan esittää puumuodossa
- Sisäkkäiset elementit muodostavat puun rakenteen
  - ulompi elementti on isäsolmu ja sisempi lapsisolmu
  - vierekkäiset elementit ovat samalla tasolla puussa, sisaruksina
  - attribuutit eivät ole lapsia, vaan sisältyvät elementtiä kuvaavaan solmuun
  - tekstisirpaleet, kommentit, käsittelyohjeet ja muut vastaavat rakennepalat esitetään myös puun solmuina
    - näillä ei voi olla lapsisolmuja

Puuesimerkki



Edellinen esimerkki XML-dokumenttina

```
<document>
  <head>Ping-Pong-palloja Hong Kongista</head>
  <ingress>...</ingress>
  <body>
    <paragraph>...</paragraph>
    <paragraph>...</paragraph>
    <picture url="..." align="left"
     text="Jaavan naavat"/>
    <paragraph>...</paragraph>
  </body>
<document>
```


Muunnokset muodosta toiseen

- Dokumentista puuksi:
  - dokumentti luetaan elementti kerrallaan ja jokaisesta elementistä muodostetaan uusi solmu puuhun
  - elementti kiinnitetään isäelementin solmuun
- Puusta dokumentiksi:
  - puu käydään läpi rekursiivisesti
    - elementin alkuosa tulostetaan ennen lapsisolmuja
    - attribuutit tulostetaan alkuosaan
    - elementin loppuosa tulostetaan lapsisolmujen jälkeen

Puurakenteen käyttö

- Lähes kaikki ohjelmistorajapinnat käyttävät XML-dokumentteja puina
  - tekstimuotoinen dokumentti on helpompi ihmiselle ja sitä on mukava siirtää koneesta toiseen, mutta siinä on paljon turhaa tietoa
  - puumuotoisen dokumentin käsittely on nopeampaa
    - kaikki operaatiot voidaan toteuttaa tietorakenteen muokkaamisina
    - ei tarvetta siirrellä tekstimassoja paikasta toiseen
- Ohjelmointirajapintoihin tutustuaan myöhemmillä luennoilla

Perustietorakenteet

Solmun käsite

- Suurin osa tietorakenteista liittyy rakenteessa oleviin tietoyksilöihin (solmut, nodes) ja niiden välisiin yhteyksiin (kaaret, arcs)
  - yleensä tietorakenne kannattaa kuvata kuvaamalla sen solmut elementeiksi
  - yhteydet syntyvät joko automaattisesti XML-dokumentin rakenteen mukana tai sitten ne koodataan erikseen
  - solmujen sisällä oleva tieto täytyy myös pystyä kuvaamaan
- Huomaa, että XML-kuvauksen ja tietorakenteen ei tarvitse sisältää täsmälleen samaa määrää informaatioita
  - Tietorakenne sarjallistetaan XML-dokumentiksi ja XML-dokumentista luodaan tietorakenne
  - Molemmissa prosesseissa ohjelma voi poistaa ja lisätä implisiittistä tietoa

Solmun tiedon esittäminen

- Tällä luennolla ei oteta erityisesti kantaa, kuinka solmun sisältämä tietotyyppi pitäisi esittää
  - esimerkeissä on käytetty yleensä primitiivityyppejä
  - normaalisti tietorakenteissa on kuitenkin monimutkaisempia tyyppejä, kuten olioita tai structeja
    - tällaiset tyypit kuvataan yleensä omina elementtijoukkoina, jotka sijoitetaan solmuelementin sisään – tällöin on järkevää sijoittaa koko tyypin kuvaus yhden elementin sisään
    - kuvauksessa täytyy olla joku järkevä syntaksi, jonka avulla tyyppi pystytään sarjallistamaan XML:ksi ja rakentamaan taas XML-koodista
    - asiaa sivutaan hieman luennon lopussa

Listat

- Listat ovat yksinkertaisia
  - listan jokainen solmu sijoitetaan elementiksi juurielementin (tai jonkun muun vastaavan isäelementin sisään)

```
<list>
  <list-item>item 1</list-item>
  <list-item>item 1</list-item>
  <list-item>item 1</list-item>
</list>
```

- Tämä toteutus ei ota kantaa, onko lista yksin- vai kaksinkertaisesti linkitetty
  - molemmat listatyypit voidaan laatia dokumentin pohjalta
- Periaatteessa listassa ei edes tarvita erillistä list-item (tms.) elementtiä, vaan tietotyyppielementit voitaisiin luetella peräkkäin

Listat viittauksilla

- Listat voidaan esittää myös yksittäisinä solmuina, jotka viittaavat toisiinsa
  - jokaisella solmuelementillä on id ja viittaus seuraavaan ja/tai edeltävään solmuun
  - solmujen ei tarvitse olla järjestyksessä, mutta tällöin jossakin täytyy kertoa ensimmäisen/viimeisen solmun id

```
<list start-id="node-1">
  <list-node id="node-7" next="node-2" ... />
  <list-node id="node-1" next="node-7" ... />
  <list-node id="node-2" ... />
</list>
```

- Tällaisen listan muokkaus XML-työkaluilla on vaikeampaa kuin edellä esitetyn

Puut

- Puut kuvataan, kuten aiemmin opittiin, normaaleina XML-dokumentteina
- Mikäli kyseessä on joukko puita, metsä, voidaan jokainen puu kuvata omana elementtinään juurielementin alla

```
<forest>
  <tree>
    <node>
      ...
    </node>
  </tree>
  <tree>
    ...
  </tree>
</forest>
```


Verkot

- Verkon syklejä ei pysty kuvaamaan suoraan XML-rakenteella
  - sykli täytyy purkaa viittaukseksi elementistä toiseen
  - eli verkko puretaan mahdollisimman hyvin puuksi ja sykliset viittaukset koodataan attribuuteilla:

```
<graph>
  <node id="node-17" ... />
    <node ... />
    ...
    <node>
      <arc ref-id="node-17"/>
    </node>
  </node>
</graph>
```

- Viittaukset täytyy koodata omiksi elementeikseen, koska niitä voi olla useita yhdestä noodista

Verkot, vaihtoehto

- Toinen tapa kuvata verkkoja on listata ensiksi kaikki verkon solmut ja antaa niille jokaiselle id
- Tämän jälkeen luetellaan solmujen väliset kaaret

```
<graph>
  <nodes>
    <node id="node-1" ... />
    ...
  </nodes>
  <arches>
    <arch node-start="node-1" node-end="node-2"/>
  </arches>
</graph>
```

- Tällainen ratkaisu mahdollistaa myös suunnattujen verkkojen kuvaamisen
  - Tapa on, kuten listojenkin tapauksessa, hieman hankalampi XML-työkaluille

Viittausten tarkistus

- Viittausten hallinta saattaa muodostua hankalaksi ja XML-dokumentista ei päällisin puolin näe, onko kaikki viittaukset kunnossa
  - Kone voi kuitenkin tarkistaa viittausten järkevyyden osittain automaattisesti
  - XML:ssa attribuutin tyypiksi voidaan määrittää ID, jolloin sen arvon täytyy olla ainutkertainen dokumentin ID-attribuuttien joukossa
    - Siis kaikkien ID-attribuuttien, ei vain samannimisten
  - Attribuutin tyyppi voi olla myös IDREF, jolloin sen täytyy viitata johonkin ID-attribuuttiin
    - Useampaan ID-attribuuttiin voidaan viitata kerralla IDREFS-tyypillä
  - Jos ehdot eivät täyty, dokumenttia ei voida validoida
  - Huomaa, että nämä säännöt eivät tarkista, että onko viittaus juuri oikeaan elementtiin
    - Tarkistus on vain tekninen

Hajautusrakenteet

- Hash Mapit ynnä muut vastaavat rakenteet puretaan esimerkiksi lista-tyyliseksi dokumentiksi, jossa hash-arvo on tallessa attribuutissa
- Periaatteessa mikä tahansa muukin kuvaus kelpaa, mutta lista on helpoin
  - Joskus saattaa olla järkevää käyttää puumaista muotoa, esimerkiksi jos hash-avaimet muodostavat hierarkian tai jos hash mapissa on useita kerroksia

Unionit ja structit

- C-kielen unionit ja structit kääntyvät helposti XML:ksi
  - Jokainen unionin ja structin tietotyyppi kuvataan omana elementtinään
    - Elementin nimi voi olla esimerkiksi käytetty tietotyyppi, muuttujan nimi talletaan attribuuttiin ja arvo elementin sisään
    - Mikäli muuttujan tyyppi on struct tai union, vastaava toimenpide suoritetaan uudestaan
  - Unionin XML-dokumenttiin syntyy vain yksi elementti, structille tulevat kaikki määritellyt

Oliot

- Oliot kuvataan vastaavalla tavalla kuin structit, eli jäsenmuuttujien arvo talletetaan
  - Structeissa ei tehdä eroa yksityisten ja julkisten muuttujien välille, joten tämä tieto täytyy lisätä – esimerkiksi attribuuttina – olion elementteihin
  - Viittaukset olioista toisiin voivat synnyttää suuria dokumentteja
    - Ne voidaan hoitaa myös ID-tyyppisinä viittauksina tai katkaista (vrt. RMI)
- Oliokielissä on valmiita kirjastoja, jotka pystyvät sarjallistamaan olioita XML-dokumenteiksi
  - Esimerkiksi Java XML Bindings (JAXB)
  - Näissä voi olla rajoituksia tietotyypeille tai luokan määrittelylle

Tietotyyppien kuvaaminen käytännössä

Tietorakenteiden yhdistely

- Yleensä yksittäinen tietorakenne per dokumentti ei ole riittävä ratkaisu
  - Sovellukset käyttävät laajoja tietorakenteita, jotka merkitsevät kokonaisuuksina, ei yksittäisinä paloina
- Suurempi tietorakenne voidaan sarjallistaa XML:ksi kahdella eri periaatteella:
  - jokainen rakenne sarjallistetaan omaksi alidokumentiksi
    - viittaukset pohjautuvat ID-attribuutteihin
  - rakenteet sarjallistetaan sisäkkäin
    - ei implisiittisiä viittauksia
    - syklit ja toistuvat viittaukset täytyy hoitaa ID-attribuuteilla

Lähteminen primitiivityypeistä

- Jokaisessa ohjelmointikielessä on joukko primitiivityyppejä, joita ei voida pilkkoa pienemmiksi
  - esimerkiksi Javassa näitä tyyppejä ovat: byte, char, short, integer, long, float, double, boolean ja class
    - merkkijono on luokka, samoin taulukot käsitellään luokkina
- Tietorakenteiden XML-muunnosten rakentaminen helpottuu, jos näille primitiivityypeille on sovittu yksiselitteinen esitystapa
  - tällöin ei tarvitse keksiä joka kerta uutta tapaa kuvata esimerkiksi kokonaislukua

Primitiivityyppiesimerkkejä

- Alla on esitetty muutamia Javan tyyppejä XML-elementteinä

```
<integer>2</integer>
<double>3.14159265</double>
<boolean>true</boolean>

<class name="java.lang.String">
  <array length="30" name="str">
    <char>T</char>
    <char>h</char>
    <char>i</char>
    ...
  </array>
</class>
```


Merkkijonot

- Merkkijonot eivät ole primitiivityyppejä kovinkaan monessa ohjelmointikielessä
  - niitä käytetään kuitenkin valtavasti
  - niiden kuvaaminen primitiivityyppien avulla on hidasta
    - ja turhaa, koska XML pohjautuu merkkijonoihin
- Ei siis kannata kikkailla ja kuvata merkkijonoa merkkitaulukkona, jossa jokainen merkki muutetaan erikseen XML-dokumentiksi
  - tämän sijaan määritellään esimerkiksi elementti <string>, joka sisältää koko merkkijonon sisällön


Kysymyksiä? Kommentteja?