Tiedon muunto ja XMI

Matti Kilpeläinen

16.10.2007

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Tässä tutkielmassa käsitellään tiedonmuuntoa UML-kaavioista XMI-muotoon ja tästä edelleen skaalautuvaksi vektorigrafiikaksi eli SVG:ksi. Tutkielmassa rakennetaan Java-pohjainen työkalu, jolla voidaan demonstroida UML-työkalujen tiedonvaihtoa XMI-muodossa ja muuntaa tieto internetissä selattavaan muotoon eli SVG:ksi. Aluksi tutkielmassa tutustutaan tutkielman kannalta kolmeen tärkeään käsitteeseen UML, XMI ja SVG.

UML muodostaa lähtökohdan monimutkaisten ohjelmistojen suunnittelulle ja XMI taas on tiedostoformaatti, jossa UML-kaaviot on muunnettu XML-tiedostoiksi ja näitä tiedostoja pitäisi voida vaihtaa eri työkalujen välillä. XMI-tiedosto voidaan muuntaa XSLT-tyylisivulla SVG:ksi, jota voidaan esittää joko nettiselaimella, jossa on SVG-plugin, tai erillisillä SVG:n näyttämiseen erikoistuneilla sovelluksilla. Tutkielmassa rakennettavan työkalun osalta keskitytään pelkästään käyttötapauskaavioiden muuntamiseen.

Tutkimuksessa tuli selväksi, että eri työkaluilla XMI:n muodostus vaihtelee vielä huomattavasti ja vaikka XMI on tarkoitettu sellaiseksi formaatiksi, jossa tiedonmuunto pitäisi olla työkalujen välillä helppoa, sitä se ei käytännössä ole johtuen eri työkalujen tavasta muodostaa diagrammeja. Tästä johtuen yleispätevän työkalun tekeminen, joka osaisi vaihtaa ja käsitellä tietoa eri UML-työkalujen välillä, on hankalaa.

ACM-luokat (ACM Computing Classification System, 1998 version): D.2.2

Avainsanat: UML, SVG, XMI

Sisältö

1 Johdanto

Suunnittelukaavioiden vaihdon (Design Interchange, DI) tarkoituksena on mahdollistaa sujuva ja näkymätön UML-standardin mukaisten dokumenttien vaihto eri UML-sovellusten välillä. Näitä UML-sovelluksia ovat kaikenlaiset UML-mallinnustyökalut, mutta näitä voivat olla myös esimerkiksi erilaiset kaavioiden piirtotyökalut, koodigeneraattorit ja erilaiset julkaisutyökalut. Myös internetissä voidaan vaihtaa ja esittää UML-malleja.

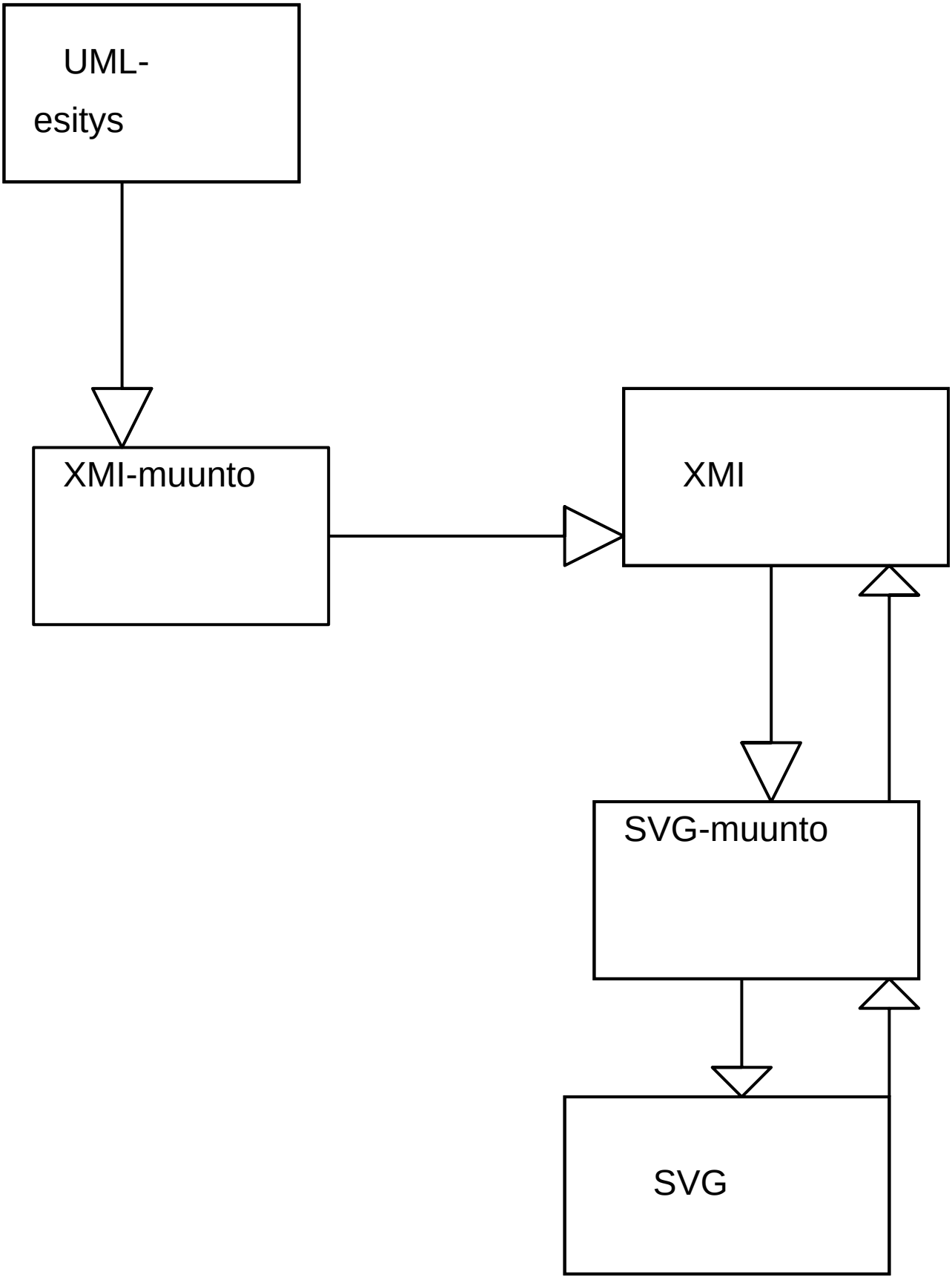
Vaikka mekanismi esittää UML-malleja oli määritelty jo UML 1.X:stä lähtien käyttäen XMI:tä, tämä mekanismi ei täysin täyttänyt mallien vaihdon tavoitteita. Ensinnäkään, se ei sisältänyt diagrammien vaihtojen tietoja. Tämä mekanismi oli vain kykenevä siirtämään informaatiota elementeistä, jotka kuuluivat UML-malliin, mutta ei tietoa siitä, kuinka nämä elementit oli esitetty ja sijoiteltu kaavioissa. Tästä syystä, jos tieto tallennetaan toisella työkalulla ja ladataan toisella työkalulla, kaikki kaaviotieto häviää. Rajoitus ei johdu XMI:stä itsestään, vaan UML-metamallista, joka ei määrittele standardia tapaa esittää kaavioiden määrityksiä.

UML-metamalliin on kehitetty ylimääräinen paketti graafista informaatiota varten jättäen malli kuitenkin täysin koskemattomaksi. Tämä paketti on yhteensopiva UML 2.0 metamallin kanssa ja myöskään tulevien muutosten vaikutusten ei pitäisi vaikuttaa UML-metamalliin. MOF-yhteensopiva metamalli on lisäys UML-metamalliin, jonka avulla saadaan XMI:n DTD:tä laajennettua. Tällä mallilla ei tapahdu tietojen menetystä työkalujen välillä Jecklen (2002) mukaan.

Kuvassa 1 on kuvattu tutkielmaa varten kehitetyn ohjelmointityökalun toiminnallinen periaate ja samalla tutkielman yleisrakenne. Työkalu suorittaa muunnon UML:n graafisesta esityksestä XMI-muotoon muunnolla, joka kuvataan luvussa 4. Lisäksi työkalu voi suorittaa muunnon SVG-muotoon.

Jotta työkalut, joissa ei ole käsitystä mallielementeistä, vaan riveistä, tekstistä ja grafiikasta toimisivat, muuntomekanismi XMI:stä SVG:ksi on kehitetty. SVG on XML-pohjainen formaatti vektorigrafiikalle, joka on hyväksytty W3C:n suosituksiin. Se sopii hyvin UML-diagrammien esittämiseen ja sitä käyttävät monet työkalut ja se tehtiin sopivaksi internetiin Lilleyn ja Jacksonin (2003) mukaan.

Tutkielman rakenne on seuraava. Aluksi käsitellään kuvauskieliä siten, että luku 2 käsittelee UML-mallinnuskieltä, XMI:tä yleisesti sekä hieman XML:ää. Luku sisältää myös skaalautuvan vektorigrafiikan määritykset. Luku 3 käsittelee tiedon muuntamista eri formaateista toiseen. Luvussa 4 esitellään muunnoksien testausta varten rakennetun Java-pohjaisen, käyttötapauskaavion muuntamiseen rajautuvan työkalun rakenne sekä vertaillaan lyhyesti muutamaa todellista mallinnustyökalua, joilla voi tehdä myös tässä tutkielmassa tarkasteltavia muunnoksia.



Kuva 1: Tutkielman yleisrakenne.

Tutkielman päättää lyhyt yhteenveto.

2 Mallinnuskielet

Mallien esittämiseen käytetään erilaisia kuvaustapoja. Kun tällainen kuvautapa on täsmällisesti määritelty, voidaan puhua mallinnuskielestä (modeling language). Mallinnuskielien esitysmuotona voidaan käyttää esimerkiksi tekstiä, taulukoita tai graafisia kaavioita. Mallinnuskieli voidaan määritellä joko formaalisti tai luonnollisella kielellä, tai käyttäen molempia Koskimiehen (2003) mukaan. Seuraavaksi esitetään mitä erilaisia kuvauskieliä tässä tutkielmassa käsitellään. Ensin esitellään mallinnuskieli UML, tämän jälkeen siirrytään esittelemään XMI ja viimeisenä käsitellään SVG.

2.1 UML

UML on teollisuusstandardi kielelle, joka määrittää, visualisoi, rakentaa ja dokumentoi ohjelmistojärjestelmien suunnittelun ja rakenteen. Se helpottaa ohjelmistoprosessia tarjoamalla mallin ohjelman rakentamiseksi. Mallinnus on välttämätön prosessi, erityisesti monimutkaisissa järjestelmissä, joka takaa täydellisyyden ja oikeellisuuden OMG:n (2007) mukaan.

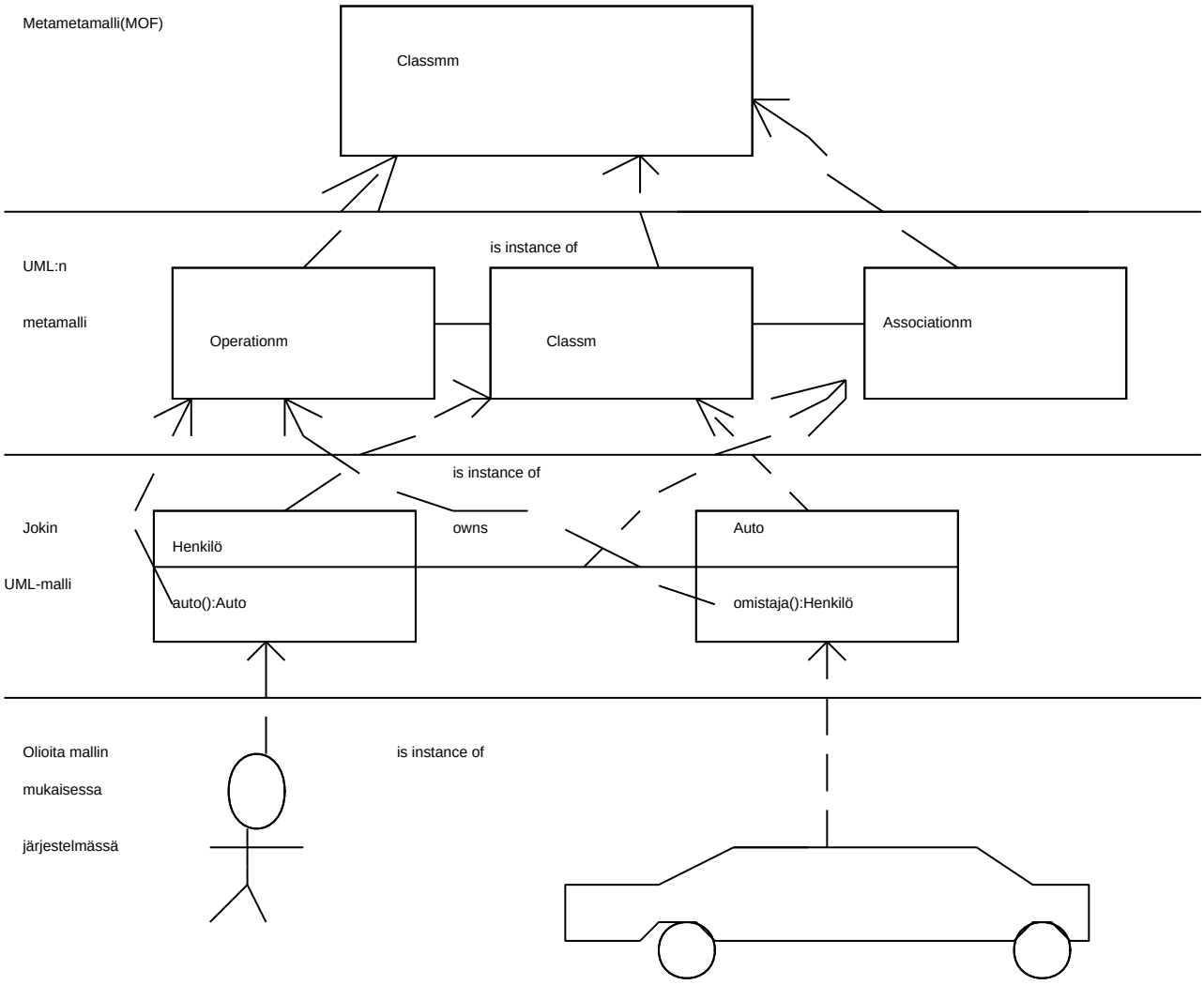
2.1.1 Yleistä UML:stä

Vuonna 1994, James Rumbaugh liittyi Grady Boochin kanssa Rational ohjelmistotaloon ja he alkoivat työskennellä yhdessä yhdistäen Boochin ja OMT:n metodeja yhdeksi. Myöhemmin, 1995, Ivar Jacobson liittyi heihin ja jatkoi heidän työtään määrityksessä määrittelemällä UML:n. Vuonna 1997 Rational lähetti heidän UML:nsä OMG:lle ehdotukseksi standardiksi. Pian tämän jälkeen OMG hyväksyi UML:n yleiseksi tietokonejärjestelmien mallinnusmääritykseksi. Nykyisin UML 1.4.2 on hyväksytty ISO-standardiksi.

Että objektit olisivat jaettavissa eri ohjelmointikielien välillä, on määriteltävä olio ottamatta huomioon ohjelmointikieltä. UML tarjoaa tähän ratkaisun tarjoten graafisen ratkaisun objektipohjaisten järjestelmien kehitykseen Elsberryn ja Elsberryn(2003) mukaan. UML:n versio 1.x tukee vain elementtien määrityksiä mallissa. Tämä ei ole riittävä graafisille työkaluille.

OMG:n metamalliarkkitehtuuri perustuu nelitasoiseen rakenteeseen kuvan 2 mukaisesti. Alimman tason rakenteen muodostavat järjestelmän ajonaikaiset oliot ja niiden väliset linkit. Kutsumme tätä tasoa järjestelmätasoksi. Seuraavalla tasolla on järjestelmän malli, esimerkiksi kuvan 3 mukaisesti luokkakaavio, jonka luokkien ilmentymistä järjestelmätaso muodostuu. Seuraavalla tasolla on vastaavasti hieman yleisempi luokkakaavio, metamalli, jonka luokkien (metaluokkien) ilmentymiä ovat

puolestaan mallitason elementit. Ylimmällä tasolla on edelleen yleisempi luokkakaavio, metametamalli (MOF, Meta Object Facility), jonka luokkien ilmentymiä ovat metamallitason elementit. Kun käytännöllisenä tavoitteena on kehys, jonka puitteissa voidaan määritellä erilaisia samaan peruskäsitteistöön pohjautuvia mallinnuskieliä, nämä neljä tasoa riittävät. Metametamalli on lopulta itsensä eräs ilmentymä Koskimiehen et al. (2003) mukaan.



Kuva 2: UML:n mallitasot (Koskimies et al., 2003).

UML 2.0 sisältää 13 diagrammityyppiä, jotka on jaettu kolmeen kategoriaan OMG:n (2003) mukaan. Kuusi diagrammityyppiä esittää sovelluksen pysyvää rakennetta, kolme esittää yleistä käyttäytymistä ja neljä esittää erilaisia näkymiä vuorovaikutukseen:

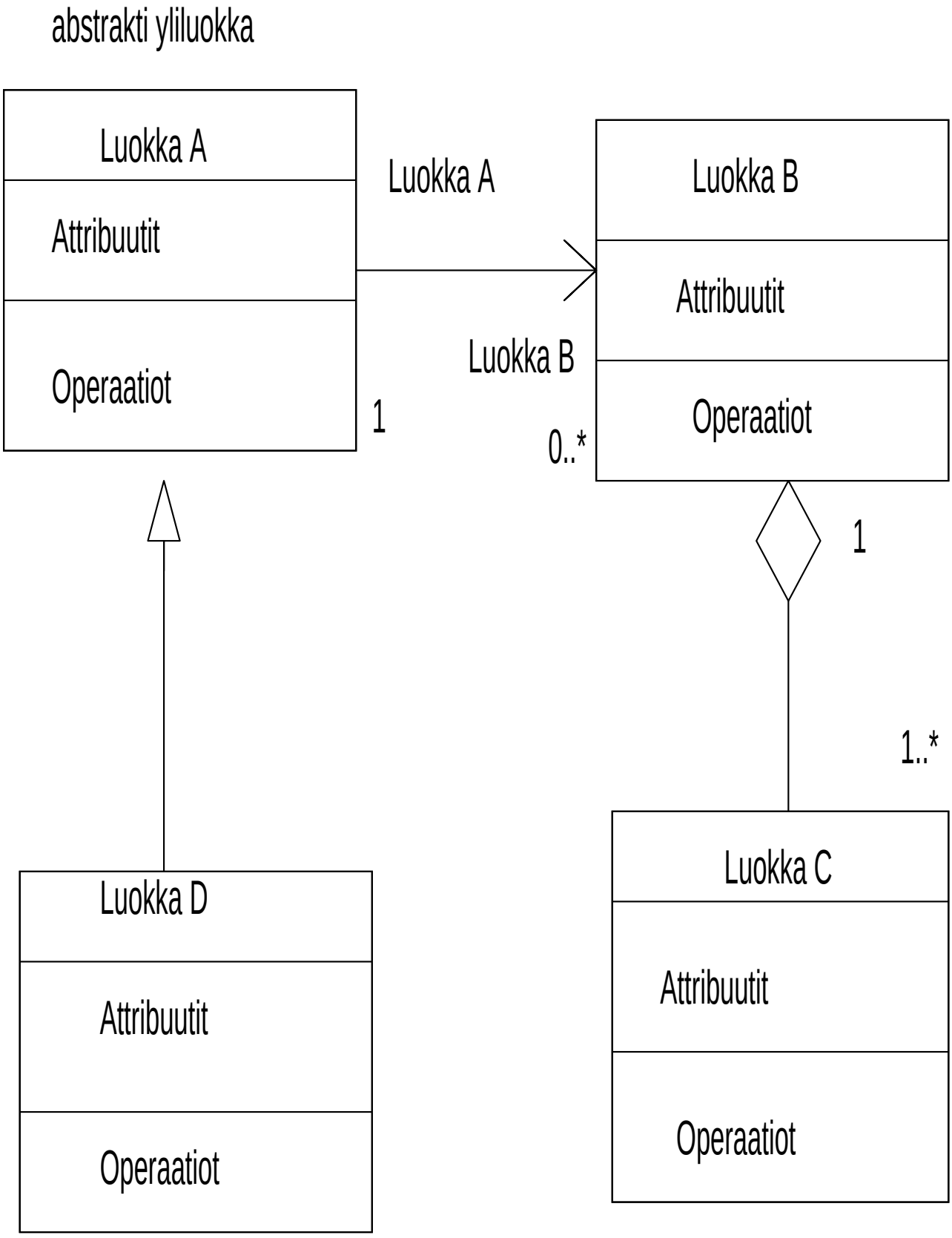
- *Rakennekaaviot:* luokkakaavio, objektikaavio, komponenttikaavio, yhdistelmäkaavio, pakettikaavio ja sijoittelukaavio.

- *Käyttäytymiskaaviot:* käyttötapauskaavio, aktiviteettikaavio ja tilakaavio.
- *Vuorovaikutuskaaviot:* sekvenssikaavio, kommunikaatiokaavio, ajoituskaavio ja vuorovaikutuksen yleiskaavio.

Seuraavana tarkastellaan kahta yleisintä UML-kaaviota: käyttötapaus ja luokkakaavio. Käyttötapauskaaviota sovelletaan tutkielmassa muunnoksia esitettäessä. Luokkakaaviota käytetään mallinnuskieliä esitettäessä.

2.1.2 Luokkakaaviot

Luokkakaavio antaa yleiskatsauksen systeemistä näyttäen sen luokat ja suhteet niiden välillä Millerin (2003) mukaan. Luokkakaaviot ovat staattisia, joten ne kuvaavat mikä vuorovaikuttaa, mutta eivät itse vuorovaikutusta. UML-luokkakaavio voi olla rakennettu esittämään elementtejä, suhteita ja XML-sanastoa visuaalisesti. UML:n luokan esitys on suorakulmio jaettuna kolmeen osaan: luokan nimeen, attribuutteihin ja operaatioihin. Abstraktien luokkien nimet ovat korostettuja kursivoidulla fontilla. Suhteet luokkien välillä ovat yhdistäviä linkkejä. Kuvassa 3 on kuvattu luokkakaavion symboleita. Luokka A on abstrakti yliluokka aliluokalle D eli luokka A on luokan D yleistys. Luokkien A ja B välillä on yksi-moneen suunnattu assosiaatio. Luokka B on luokan C koosteluokka. Luokkakaaviossa on kolmenlaisia suhteita, joita ovat assosiaatio, kooste ja yleistys. Assosiaatiolle on annettu roolinimet luokka A ja luokka B. Kooste on assosiaatio, jossa jokin luokka kuuluu kokoelmaan. Koosteella on timantti loppupäässä osoittamassa osaa, joka sisältää kokonaisuuden.



Kuva 3: Luokkakaavion symboleita.

Peruselementit UML:n luokkakaaviossa ovat seuraavat Millerin (2003) mukaan:

- *Luokka:*

  Luokka esittää koosteena rakenteellisia piirteitä ja määrittää nimiavaruuden noille piirrenimille. Tästä johtuen, kaksi luokkaa voivat sisältää attribuutin nimeltään nimi, mutta niiden luokkien nimiavaruudet tekevät näistä kahdesta attribuutista erottuvat.

- *Attribuutti:*

  Kukin luokka voi vapaasti määritellä joukon attribuutteja. Kullakin attribuutilla on oma tyyppinsä, esimerkiksi merkkijono string, double ja float viittaavat sisäänrakennettuihin datatyyppeihin, jotka on määritelty XML-skeeman määrityksessä.

- *Operaatio:*

  Operaatio luokassa määrittää osan luokan käyttäytymistä. Tämä operaatio ei oleta mitään parametreja, mutta ne voisivat olla sulkujen välissä. Operaatioita ei käytetä XML-sanaston määrityksessä, mutta niiden määritys olisi kriittinen verkkopalveluille, esimerkiksi WSDL-määritykselle SOAP-viesteissä.

- *Assosiaatio:*

  Assosiaatio yhdistää kaksi tai monta luokkaa mallissa. Jos assosiaatiolla on nuoli vain toiseen suuntaan mallissa, tämä tarkoittaa, että viestit kulkevat vain yhteen suuntaan. Assosiaatio on suhde kahden instanssin kesken kahdessa luokassa. Kahden luokan välillä on assosiaatio, jos toisen luokan tarvitsee tietää jotain toisesta toimiakseen.

- *Rooli ja kertautuminen:*

  Yhteyden päätepiste voi määrittää luokan roolin. Lisäksi, assosiaation päihin liittyy lukumäärä-suhde. Luokkien välinen suhde voi olla taulukon 1 mukainen.

- *Yleistys:*

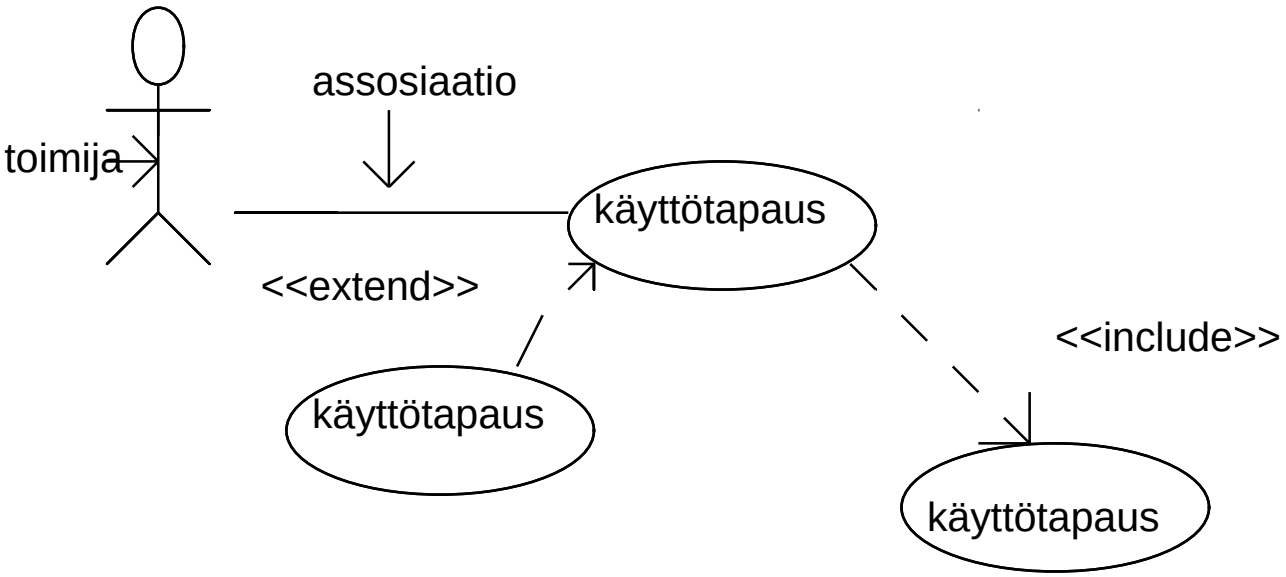
  Yleistys on perustavaa laatua oleva objektisuuntautunut malli olioluokkien periyttämiseksi. Periytymislinkki osoittaa yhtä luokkaa, joka on yliluokka toiselle. Yleistyksessä on kolmio osoittamassa yliluokkaa.

Assosiaatiolla on kaksi päätepistettä. Päätepisteessä voi olla jokin roolinimi kuvaamaan assosiaation luonnetta. Suunnistettavuusnuoli assosiaatiossa näyttää mihin suuntaan assosiaatiossa voidaan suunnistaa tai kysellä. Nuoli myös ilmaisee kuka omistaa assosiaation toteutuksen. Assosiaatiot, joilla ei

ole suunnistettavuusnuolia ovat kaksisuuntaisia. Yhteyden loppupään kertautuminen on mahdollisten instanssien lukumäärä. Kertautumiset ovat yksittäisiä arvoja tai numerorajoja. Jokaisessa luokkakaaviossa on luokkia, yhteyksiä ja kertautumisia. Viestien suunta ja roolit ovat vapaaehtoisia Millerin (2003) mukaan.

2.1.3 Käyttötapauskaaviot

Käyttötapauskaaviot kuvaavat mitä järjestelmä tekee ulkopuolisen tarkkailijan näkökulmasta. Painotus on paremminkin sillä, mitä systeemi tekee kuin miten. Käyttötapauskaaviot ovat läheisesti kytketty skenaarioihin. Skenaario on esimerkki mitä tapahtuu, kun joku on vuorovaikutuksessa järjestelmän kanssa. Kuvassa 4 on kuvattu käyttötapaus kaavion symbolit Millerin (2003) mukaan.



Kuva 4: Käyttötapauskaavion symbolit.

Taulukko 1: Erilaiset lukumääräsuhteet.

Lukumäärä	Merkitys
0..1	nolla tai yksi ilmentymä.
n..m	tarkoittaa n:stä m:n ilmentymää.
$0..*\ tai *$	ei mitään rajaa ilmentymille (mukaan lukien nolla).
1	tarkalleen yksi ilmentymä.
$1..*$	vähintään yksi ilmentymä.

Käyttötapaus on yhteenveto skenaarioista, joilla on tietty yksittäinen tehtävä tai tavoite. Toimija on se kuka tai mikä käynnistää tapahtumat, jotka liittyvät tiettyyn tehtävään. Toimijat ovat yksinkertaisesti rooleja, joita ihmiset tai objektit tekevät. Toimijan ja käyttötapauksen välinen suhde on assosiaatio. Toimijat ovat tikku-ukkoja. Käyttötapaukset ovat ovaaleja. Assosiaatiot on viivoja, jotka liittävät toimijat käyttötapauksiin. Käyttötapauskaavio on kokoelma toimijoita, käyttötapauksia ja niiden välisiä yhteyksiä. Yksittäisellä käyttötapauksella voi olla useita toimijoita.

Suhde include on sellainen suhde, jossa toinen käyttötapaus sisältää toisen käyttötapauksen toiminnallisuuden. Include-suhde tukee toimintojen uudelleenkäyttöä käyttötapausmallissa.

Include-käyttötapausta käytetään silloin, kun käyttötapaus on yleinen kahdelle tai useammalle käyttötapaukselle tai jos käyttäytymisen tulos, jonka sisältävä käyttötapaus määrittää, on tärkeä peruskäyttötapaukselle.

Include-käyttötapaus kuvataan diagrammissa yhtenäisellä viivalla avoin nuoli osoittamassa peruskäyttötapauksesta sisältyvään käyttötapaukseen. Avainsana «include» on kytketty katkoviivalla esitettävään suhteeseen IBM corporationin (2005) mukaan.

Extend-suhdetta käytetään määrittämään, että yksi käyttötapaus laajentaa toisen käyttötapauksen käyttäytymistä.

Extend-suhde määrittää, että laajentavan käyttötapauksen yhdistäminen on riippuvainen siitä, mitä tapahtuu kun peruskäyttötapaus suoritetaan. Yhdelle peruskäyttötapaukselle voidaan määrittää useita laajennussuhteita.

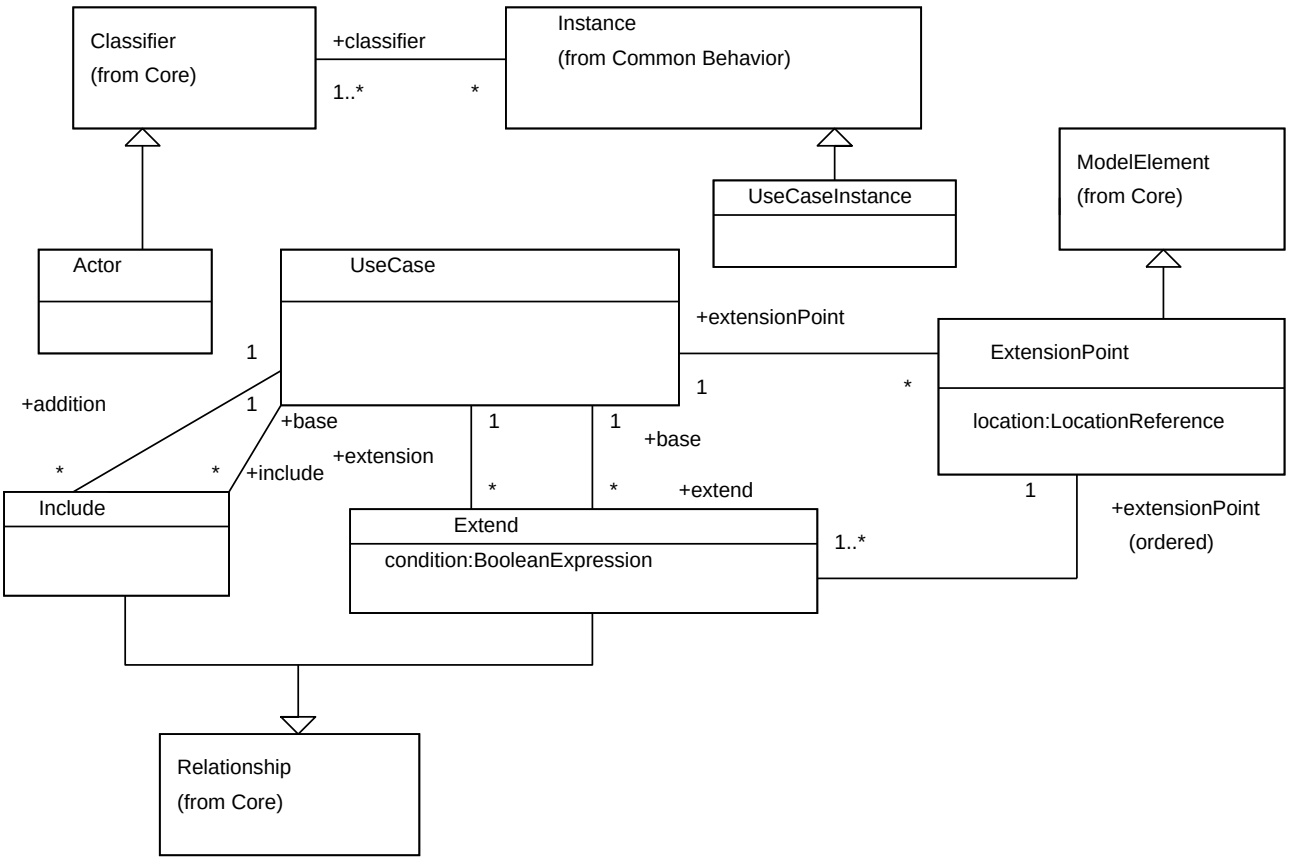
Extend-suhteita voidaan käyttää mallissa esimerkiksi silloin, kun käyttötapaus on valinnainen järjestelmän käyttäytymiselle ja aliohjelma suoritetaan vain jossain tietyissä tapauksissa.

Extend-suhde on kuvattu katkoviivalla avoin nuolenkärki osoittamassa laajentavasta käyttötapauksesta peruskäyttötapaukseen. Nuoli on nimetty avainsanalla «extend» IBM corporationin (2005) mukaan. Peruskäyttötapaukselle voidaan määrittää laajennuspiste kullekin laajentavalle käyttötapaukselle.

Kuvassa 5 on kuvattu käyttötapauskaavion metamallitason elementit.

2.1.4 Metamallin laajennus

UML-metamallin laajennus kuvaa metamallin diagrammitiedolle(DI metamodel), johon diagrammien vaihtomekanismi perustuu Gentleware AG:n et al. (2003) mukaan. Se on laajennus UML-metamalliin



Kuva 5: Käyttötapauskaavion metamallin elementit.

ja perustuu UML 1.4:ään. XMI[UML] mallienvaihto sisältää vain mallin tiedon, muttei graafista informaatiota. Diagrammien vaihtolaajennus sallii graafisen informaation sisällyttämisen diagrammeihin, joita käytetään UML-malleissa. Tämä mahdollistaa myös XMI:n laajentamisen.

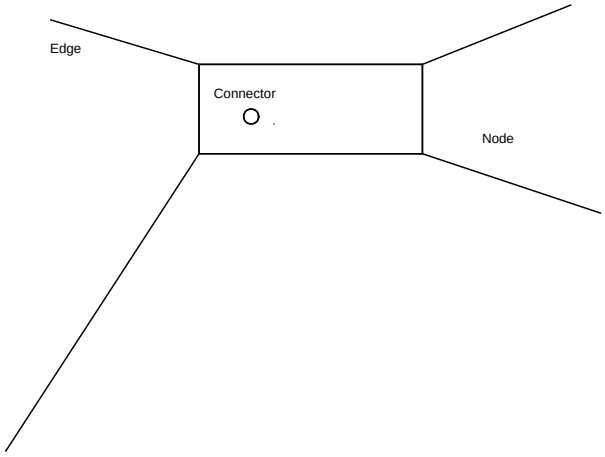
Tämä laajennus lisää uuden paketin UML-metamallipaketteihin. Kuitenkaan olemassa olevaa standardia ei muuteta millään tavalla. Ehdotettu laajennus ja UML-metamalli pidetään mahdollisimman riippumattomina, jotta ainoastaan linkit laajennuksesta UML-metamalliin sisällytetään. Täten, graafinen ja mallien informaatio on selvästi erotettu toisistaan.

Ehdotettu paketti sisältää elementit, jotka kuvaavat diagrammi-informaation kaikille UML:n kaavioelementeille. Työkalukohtaisia laajennuksia voidaan määrittää lisäpaketeissa. Esimerkiksi, jos työkalu lisää piirtomahdollisuuksia erilaisille uusille muodoille, nämä voidaan tarjota. Tämän tarkoituksena on taata, että työkalut, jotka eivät tue ylimääräisiä laajennuksia, eivät myöskään luo näitä muotoja yksinkertaisesti hylkäämällä näiden pakettien sisältämän informaation.

Metamallilaajennus perustuu ideaan mallintaa UML-diagrammien sisältöä graafeina. Ydinluokat ovat GraphNode ja GraphEdge, joilla näkyvät mallielementit esitetään. Kantaluokka graafisille elementeille on GraphElement. Graafielementit linkitetään toisiinsa luokalla, jota kutsutaan GraphConnectoriksi. Tämä mahdollistaa GraphEdgen linkityksen GraphNodeen tai toiseen GraphEdgeen. GraphConnector ei salli kahta GraphNodea linkitettäväksi. GraphElement voi omistaa minkä tahansa määrän GraphConnectoreita, joita kutsutaan ankkuripaikoiksi. Ne mahdollistavat minkä tahansa määrän GraphEdgejä olevan niihin kytkeytyneinä. GraphEdgeen viittaa kaksi GraphConnectoria, jotka ovat sen päätepisteet. GraphEdgen näkökulmasta näitä kutsutaan ankkureiksi. Kaksi GraphConnectoria on järjestetty samalla tavalla kuin vastaavien GraphEdgien kulkupisteet.

Ensimmäinen GraphConnector vastaa ensimmäisestä GraphEdgen kulkupisteestä ja toinen GraphConnector vastaa viimeisestä kulkupisteestä. GraphConnectorin ei tarvitse saada samaa positioita, joihin GraphEdgien päätepisteet viittaavat. Kun useampi kuin yksi GraphEdge viittaa siihen, se palvelee virtuaalisena kokoelmana pisteitä GraphEdgeille, esimerkiksi keskellä solmua. Kaikki linkitetyt GraphEdget osoittavat suoraan GraphConnectoriin mutta sen kulkupisteet voivat esimerkiksi olla määritetty päättyvän solmun rajalle. Kuvassa 6 on esimerkki GraphConnectorista, johon liittyy säteittäiset kaaret.

Assosiaatiosäiliö mahdollistaa sisäkkäisten elementtien hierarkian rakentamisen. Kunkin GraphElementin on mahdollista sisältää rajoittamaton määrä DiagramElementtejä, joten se voisi sisältää kokonaisen aligraafin. Sisällytyshierarkia on erityisen käyttökelpoinen, kun mallinnetaan monimutkaisia esityksiä mallinnuselementeillä.



Kuva 6: GraphConnector ja säteittäiset kaaret (Gentleware et al.,2003)

Esimerkiksi luokat esitetään GraphNodella, joka omistaa sisältyvät GraphNodet operaatio-osalle ja attribuuttiosalle. Attribuuttiosa omistaa edelleen sisällytetyt GraphNodet, jotka esittävät attribuutteja.

Attribuutin GraphNode taas sisältää DiagramElementtejä, jotka esittävät osia attribuutista, kuten näkyvyys, nimi, tyyppi tai oletusarvo. Kuva 7 esittää yksinkertaista luokkaa, jossa on sisältyviä graafielementtejä, jotka ovat GraphNodeja tässä tapauksessa.

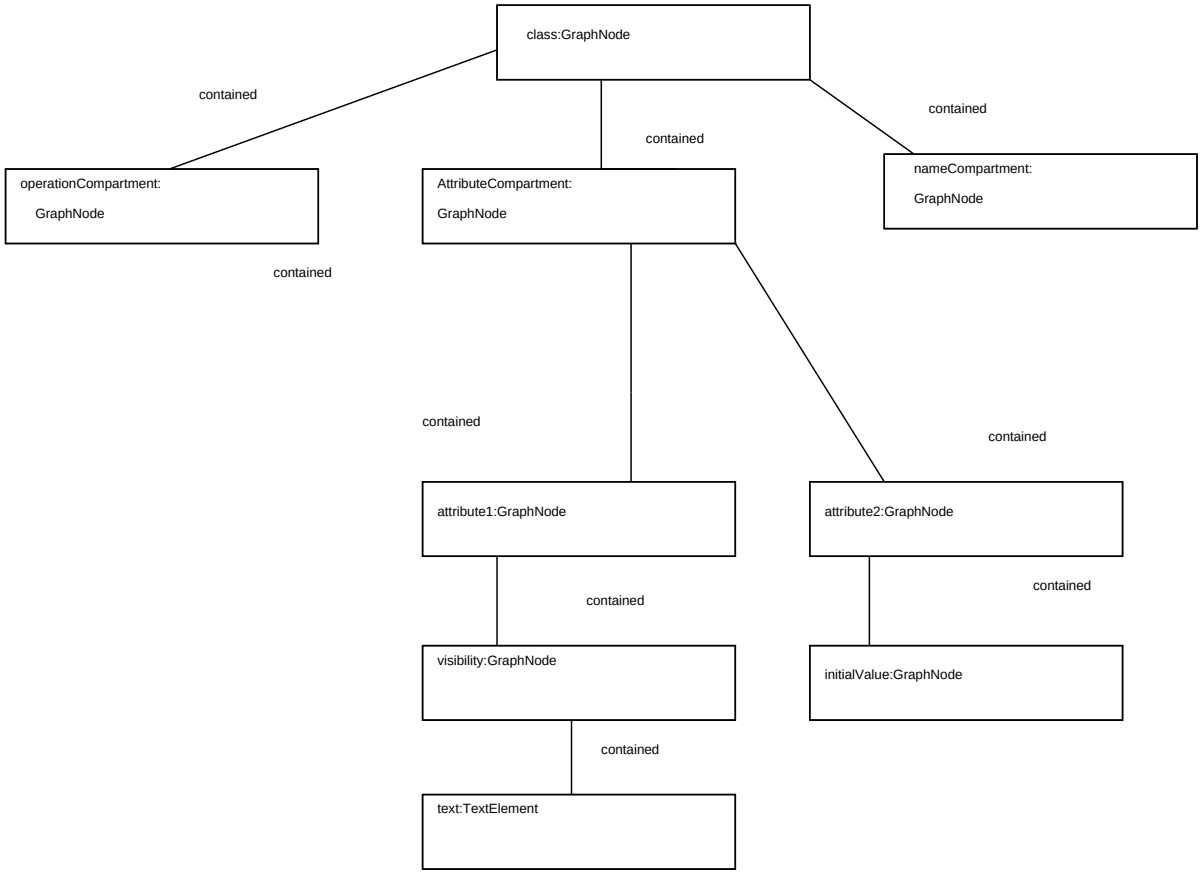
GraphElementin graafinen esitys on laajasti riippuvainen semanttisesta mallista eikä nimenomaisesti tallennettu DI-metamalliin ylimäärän välttämiseksi. Esimerkiksi tietoa siitä, että luokka piirretään suorakulmiona ei ole tallennettu DI-metamalliin. Kuitenkin sen positio ja koko tallennetaan tarvittaessa. Piirtotyökalulla ja XSL-tyylisivulla täytyy olla semanttinen tieto piirtää nämä elementit oikein.

LeafElementti eli lehtielementti voi olla TextElement, joka näyttää yksinkertaista tekstiä, GraphicPrimitive elementti, joka näyttää yksinkertaisia piirroksia, kuten ellipsejä tai UriResource, joka näyttää esim. kuvia ulkoisesta lähteestä.

TextElementtejä käytetään esittämään attribuuttien osia, operaatioita, nimiä ja muita tekstejä, jotka ovat osa mallielementtejä. Esimerkiksi attribuutin näkyvyys voi olla tekstiä, kuten public tai +. Tämä teksti on tallennettu attribuuttiin text TextElementissä.

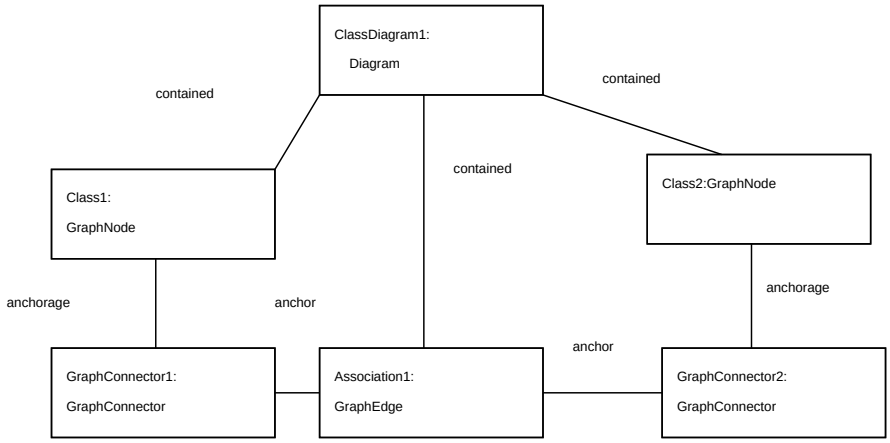
Diagram on erityinen solmu. Se on kaikkein ylimmäinen GraphElement mille tahansa graafille tai diagrammille tässä terminologiassa ja rekursiivisesti sisältää kaikki muut graafiset elementit. Diagrammilla on nimi ja ikkuna. Ikkuna on piste, joka ilmaisee nykyisen näkyvän diagrammin osan vasemman yläkulman. Sillä on myös zoomaustekijä, joka sallii sen näkymisen erilaisilla asteikoilla.

Yksinkertainen esimerkki diagrammista on annettu kuvassa 8. Se näyttää luokkadiagrammin, jossa



Kuva 7: Esimerkki sisälletyistä GraphElementeistä (Gentleware et al., 2003).

kaksi luokkaa on kytketty toisiinsa assosiaatiolla. Diagrammi ja niiden luokat on kuvattu GraphNode-solmujen avulla, assosiaatiota edustaa GraphEdge. Tämä objektidiagrammi näyttää myös roolit , jotka GraphConnectorit olettavat linkittäessään GraphNodeja ja GraphEdgejä.



Kuva 8: Diagram-elementti, jossa on kaksi luokkaa.

Diagrammielementtien (DiagramElement) ulkonäkö on määritelty ominaisuusattribuutilla. Taulukko 2 määrittää standardin joukon valinnaisia ominaisuuksia. Se määrittelee fonttiperheen ja koon kuten myös viivojen tyylin, paksuuden ja värin. Kukin asetettu ominaisuus ylikirjoittaa olemassaolevan saman tyypin GraphElementissä.

Taulukko 2: Standardoidut diagrammielementtien ominaisuudet.

Avain	*Tyyppi*	*Esimerkki*	*Kuvaus*
FontFamily	string	Times, Courier	fontin nimi
FontSize	double	11.0	fontin koko pikseleinä
LineStyle	string	kiinteä, katkonainen	viivan tyyli
Viivanpaksuus	double	2.0	viivanpaksuus pikseleinä
FontColor, ForegroundColor, BackgroundColor	integer	FF00FF sinipunaiselle	24-bittinen väriarvo RGB formaatissa

Taulukko 2: Standardoidut ominaisuudet (jatk.).

Translucent	boolean	totta, epätotta	

Jos ominaisuutta ei ole asetettu, DiagramElementti käyttää hyväkseen säiliön GraphElement ominaisuuksia.

Kaavioidenvaihtolaajennuksessa käytettävä koordinaattijärjestelmä määrittää, että x-akseli osoittaa itään ja y-akseli osoittaa etelään. Positio ja koko määritellään double-arvoilla, jotka käyttävät pikseleitä mittayksikkönä. Liukulukutyyppi double sallii osapikselin tarkkuuden. Tämä estää mahdollisen informaation menetyksen, kun skaalataan tai muuten manipuloidaan diagrammeja.

Yleisesti ottaen käytetyt positioarvot ovat suhteellisia, toisin sanoen, mikä tahansa GraphElementin positioarvo on suhteessa sitä ympäröivään säiliöön. Tämä tarkoittaa sitä, että säilön positio on lähtökohta kunkin lapsen sijainnille. Saavuttaakseen absoluuttisen position DiagramElementille, on pakollista navigoida rekursiivisesti läpi säiliön GraphElementtien, kunnes GrahpElementti, jolla ei ole säiliötä eli juuri, saavutetaan. Tämän juurisolmun täytyy olla diagrammi, koska vain diagrammilla ei ole säiliötä.

Diagrammin positio on (0,0) oletuksena. Ikkuna määrittää vasemman yläkulman koordinaatin Diagrammissa, jota parhaillaan tarkastellaan.

Suhteelliset koordinaatit sallivat hierarkiaan sisällytettyjen solmujen siirron vaihtamalla ylimmän tason solmun positiota. Esimerkiksi luokan siirto voi tapahtua siirtämällä GraphNodea, joka esittää sitä. Kaikki GraphElementit, jotka sisältyvät siihen luokkaan, siirretään automaattisesti, koska ne on sijoitettu suhteellisesti siihen nähden.

2.2 XMI

2.2.1 Yleistä XMI:stä

XML eli laajennettu merkitsemiskieli kuvaa luokan tieto- olioita, joita kutsutaan XML-dokumenteiksi ja tämä kieli kuvaa osittain myös tietokoneohjelmien käyttäytymisen, jotka käsittelevät näitä dokumentteja. XML on sovellusprofiili tai rajoitettu muoto SGML:stä, standardista yleisestä merkitsemiskielestä. Perusrakenteeltaan XML- dokumentit muodostuvat SGML-dokumenteista H W3C XML Core Working Groupin (2006) mukaan.

XML:n täytyy olla hyvin muodostettua eli siinä täytyy olla vastaava lopetustagi kullekin aloitustagille. DTD (Document type definition)-määritysten avulla luodaan sääntöjä XML-dokumentin muodostamiseen. Tällä tiedostolla voidaan tarkistaa, onko XML-tiedosto validi. Ei-validoiva jäsentäjä tarkastaa XML-syntaksin vain XML:n ydinsyntaksia vasten, kun taas validoiva jäsentäjä tarkastaa myös syntaksin DTD-tiedostoa vasten. Nimiavaruudet ja skeemat mahdollistavat XML-tiedostojen rakennuksen ohi DTD:n rajoitteiden. Muilta lainattujen sanastojen käyttöön on kehitetty XML-nimiavaruudet. DTD:t eivät anna käyttää nimiavaruuksia toisin kuin skeemat Andersonin et al. (2000) mukaan.

XMI on metadatan vaihtostandardi, joka mahdollistaa objektien esityksen käyttäen XML:ää, joka on yleinen tiedon esitystapa www:ssä. XMI tarjoaa standardin esittää objekteja XML:llä mahdollistaen tehokkaan objektien vaihdon käyttäen XML:ää. XMI määrittelee kuinka XML- määrityksiä luodaan malleista. XMI on pakollista, koska XML ei ole objektipohjaista ja objektit on mapattava XML:ään.

XMI on läheisesti liitoksissa mallinnusstandardeihin, mahdollistaen käyttää mallinnusta XML:n rakennuksessa. XMI 2.0 määrittelee, kuinka luoda XML-määrityksiä mallista ja edelliset versiot, kuinka XML-tyypit on määritelty mallissa. Sekä määritykset että DTD:t määrittelevät XML-dokumentin sisällön. Lisäksi XMI määrittelee, kuinka malleista, DTD:istä ja skeemoista voidaan käänteisesti tehdä XML-dokumentteja.

Version 2.1 mukaiset XML:n prosessointikäskyt ovat unix-merkkijärjestelmässä seuraavaa muotoa:

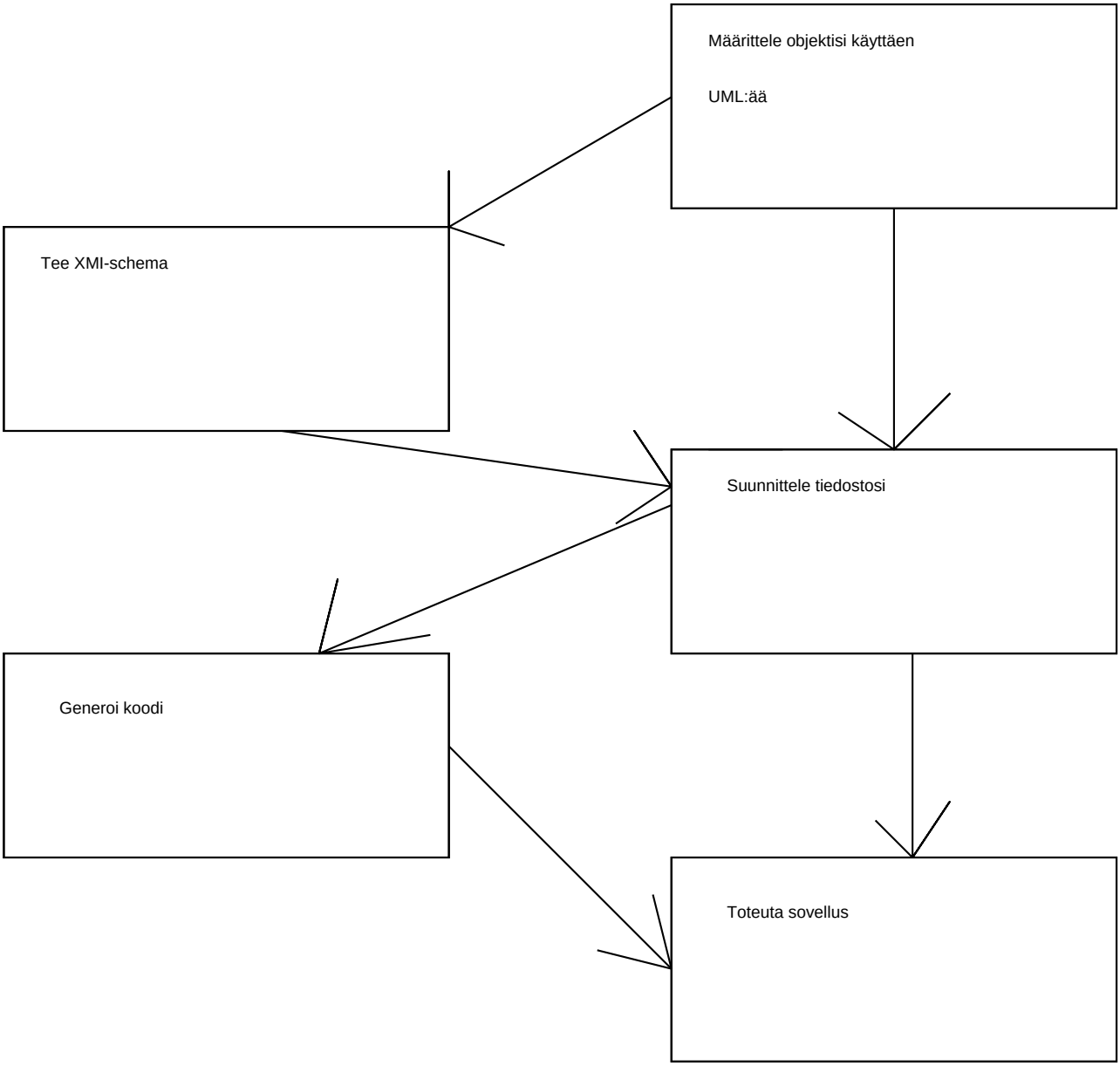
<?xml version="1.0"encoding="UTF-8»

XMI-elementtiä käytetään yleensä dokumentin juurielementtinä. Tällä on versioattribuutti ja ja attribuutin täytyy käyttää nimiavaruusetuliitettä XMI-nimiavaruudelle. Tässä on tyhjän XMI-dokumentin määrittely:

<xmi:XMI xmi:version="2.1"xmlns:xmi="http://www.omg.org/XMI»

XML:n id-attribuutilla on tyyppi ID, joten sen arvon täytyy olla laillinen tunniste XML-elementille. Id-attribuutin arvon täytyy olla yksilöllinen kullekin dokumentille, se ei vielä takaa sen yksilöllisyyttä dokumenttien joukossa. Sitä käytetään määrittämään suhteita objektien välillä Grosen (2001) mukaan. Viittaus on kahden objektin välinen suhde. XML-attribuutilla (assosiaation pää) on tyyppi IDREFS. Attribuutin arvo koostuu tunnisteista XML-elementeille, jotka vastaavat viitattuja objekteja erotettuna tyhjillä väleillä. Tyyppiä ID on havainnollistettu kuvassa 32. IDREFS tulee esille kuvassa 13 ja vastaava viittaus esimerkiksi kuvassa 27 ja liitteessä 7.

Kuvassa 9 on esitetty miten objektipohjaisia UML-kaavioita voidaan muuntaa XMI-muotoon.



Kuva 9: XMI-prosessi Grosen et al. (2001) mukaan.

Kaksi kuvan 9 prosessin vaiheista ovat vapaaehtoisia. XMI ei vaadi käyttämään XML-validointia sovelluksissa. Koodin luonti mallista on myös valinnaista, koska sen voi toteuttaa halutulla tavalla.

2.2.2 XMI-malli

XMI-malli kuvataan kolmella luokalla: XMI, Extension ja Documentation. Ylimmän tason XML-elementti XML-dokumentille, joka sisältää vain XMI-dataa, on XMI-elementti. Sen määritelmä on esitetty kuvassa 10.

```
<xsd:complexType name="XMI"> <xsd:choice minOccurs="0"
maxOccurs="unbounded">
<xsd:any procesContents="strict"/> </xsd:choice>
<xsd:attribute ref="id"/>
<xsd:attributeGroup ref="identifyAttribs"/>
<xsd:attrubuteGroup ref="LinkAttribs"/>
<xsd:attribute name="tyyppi" type="xsd:QName"
use="optional" form="qualified"/>
<xsd:attribute name="version" type="xsd:string"
fixed="2.0" use="optional" form="qualified"/>
</xsd:complexType><xsd:element name="XMI" type="XMI"/>
```

Kuva 10: XMI-elementin määritelmä (OMG, 2003).

Jos XMI:n versioattribuutti on mukana, sen täytyy olla yhteensopiva XMI-määritysten versionumeroiden kanssa. Se osoittaa, että metadata sisältää sen version kanssa yhteensopivan XMI-määrityksen. Versioattribuutti on valinnainen, koska kullakin XMI-versiolla on yksilöllinen nimiavaruusnimi muotoa "http://schema.omg.org/spec/XMI/version"OMG:n (2003) mukaan.

XMI-elementin ei tarvitse olla XML-dokumentin juurielementti, sen voi sisällyttää sen minkä tahansa XML-elementin sisälle, jota ei ole sarjallistettu tämän määrityksen mukaan. Jos dokumentti sisältää vain XMI-tietoa, XMI-elementti ei ole tyypillisesti läsnä, kun on vain yksi ylimmän tason objekti. XMI:n versioattribuuttia käytetään XMI-tiedon alkamiseen ja XMI-version tunnistamiseen, kun XMI-elementti ei ole itse läsnä.

XMI-mallin laajennusluokka (extension class) on suunniteltu sisältämään laajennettu tieto käyttäjän mallin ulkopuolelta. Laajennukset ovat XMI-luokan moniarvoinen attribuutti ja voivat olla myös XMI-dokumentissa. Skeema tälle laajennokselle on kuvattu kuvassa 11. Laajennoksen käyttöä Enterprise Architect-välineessä on havainnollistettu kuvassa 33.

```
<xsd:complexType name="Extension"> <xsd:choice minOccurs="0"
maxOccurs="unbounded">
<xsd:any procesContents="lax"/> </xsd:choice>

<xsd:attribute ref="id"/>
<xsd:attributeGroup ref="ObjectAttribs"/>
<xsd:attribute name="extender" type="xsd:string"
use="optional"/>
<xsd:attribute name="extenderID" type="xsd:string"
 use="optional"/>
</xsd:complexType>
<xsd:element name="Extension" type="Extension"/>
```

Kuva 11: Laajennusluokan skeema (OMG, 2003).

Attribuutin extender pitäisi kuvata mikä työkalu loi laajennuksen. Tarkoitus on, että työkalut voivat olla huomioimatta laajennoksia, jotka on tehty muilla työkaluilla, ennen kuin laajennuselementin sisältöä käsitellään. ExtenderID on vaihtoehtoinen sisäiselle ID:lle laajennustyökaluista. Muut attribuutit sallivat yksilöllisten laajennusten tunnistamisen ja käyttämisen välittäjinä paikallisille ja ulkoisille laajennuksille.

Documentation-luokka (kuva 12) sisältää tietoa XMI-dokumentista eli esimerkiksi dokumentin omistaja, kontaktihenkilö, lyhyet ja pitkät kuvaukset dokumentista, työkalu, jolla dokumentti luotiin, työkalun versio, tekijänoikeudet tai muut laillisuusmerkinnät koskien dokumenttia. Kaikkien attribuuttien datatyyppi dokumentissa on String. Documentation-luokan käyttö ilmenee kuvan 31 XMI-dokumentista.

XMI-malli mahdollistaa mallien välisten erojen (differences) esittämisen lisäyksinä (add), poistoina (delete) ja korvauksina (replace). Add-luokka esittää lisäyksen kohdejoukkoon objekteja tähän dokumenttiin tai muihin dokumentteihin. Sijaintiattribuutti (position) ilmaisee mihin lisäys kohdistuu suhteessa muihin XML-elementteihin. Oletusarvona , -1, ilmaisee uuden elementin lisäyksen kohdeelementin loppuun. Lisäysattribuutti (addition) viittaa joukkoon objekteja, jotka lisätään.

```
<xsd:complexType name="Documentation"> <xsd:choice minOccurs="0"
maxOccurs="unbounded">

<xsd:element name="contact" type="xsd:string"/>
<xsd:element name="exporter" type="xsd:string"/>
<xsd:element name="exporterVersion" type="xsd:string"/>
<xsd:element name="longDescription" type="xsd:string"/>
<xsd:element name="shortDescription" type="xsd:string"/>
<xsd:element name="notice" type="xsd:string"/>
<xsd:element name="owner" type="xsd:string"/>
<xsd:element ref="Extension"></xsd:choice>
<xsd:attribute ref="id"/>
<xsd:attributeGroup ref="ObjectAttribs"/>
<xsd:attribute name="contact" type="xsd:string"
 use="optional">
<xsd:attribute name="exporter" type="xsd:string"
use="optional"/>
<xsd:attribute name="exporterVersion"
type="xsd:string" use="optional">

<xsd:attribute name=""longDescription" type="xsd:string"
 use="optional"/>
<xsd:attrubute name="shortDescription" type="xsd:string
 use="optional">
<xsd:attribute name="notice" type="xsd:string"
use="optional"
/>
<xsd:attribute name="owner" type="xsd:string"
use="optional"/></xsd:complexType>
</xsd:element name="Documentation" type="Documentation"/>
```

Kuva 12: Documentation-luokan skeema (OMG, 2003).

Replace- luokka esittää kohdeobjektiryhmän poistamisen ja objektien lisäyksen, joihin on viitattu replacement-attribuutissa. Sijaintiattribuutti ilmaisee kuinka sijoittaa korvaus suhteessa toisiin XML-elementteihin. Oletusarvo -1 ilmaisee korvaavan elementin sijoittumisen kohde-elementin loppuun. Korvausattribuutti (replacement) viittaa objektiin, joka korvaa kohde-elementin.

Delete-luokka esittää kohdeolioryhmän poistamista tässä tai muissa dokumenteissa. Difference-luokka on yliluokka lisäys-, korvaus- ja poistoluokille. Näiden luokkien määritykset on kuvattu kuvassa 13.

```
<xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element nimi="target"> <xsd:complexType>
<xsd:choice minOccurs="0" maxOccurs="unbounded"
<xsd:any proces Contents="skip"/>
</xsd:choice><xsd:anyAttribute process Contents="skip" />
</xsd:complexType>
</xsd:element> <xsd:element name="difference" type="Difference"/>
<xsd:element name="container" type="Difference"/>
<xsd:element ref="Extension"/>
</xsd:choice><xsd:attribute ref="id" />
<xsd:attributeGroup ref="ObjectAttribs"/>
<xsd:attribute name="target"
  type="IDREFS" use="optional"/>
<xsd:attribute name="container" type="xs:IDREFS" use="optional"/>
</xsd:complexType><xsd:element name="Difference"
  type="Difference"/><xsd:complexType name="Add">
<xsd:complexContent><xsd:extension base="Difference">
<xsd:attribute name="position" type="xsd:string" use="optional">
  <xsd:attribute name="addition" type="xsd:IDREFS" use="optional"/>
</xsd:extension></xsd:complexType><xsd:element nimi="Add"
  type="Add"/><xsd:complexType name="Replace">
<xsd:complexContent><xsd:extension base="Difference">
  <xsd:attribute name="position" type="xsd:string" use="optional">
</xsd:attribute name="replacement" type="xsd:IDREFS" use="optional"/>
</xsd:extension></xsd:complexContent>
  </xsd:complexType><xsd:element name="Replace" type="Replace"
<xsd:complexType name="Delete"><xsd:complexContent>
<xsd:extension base="Difference"/>
  </xsd:complexContent></xsd:complexType>
<xsd:element name="Delete" type="Delete"/>
```

Kuva 13: Lisäys-, korvaus- ja poistoluokkien määritys (OMG,2003).

2.2.3 Skeeman XMI-attribuutit

Tässä on kuvattu eräitä tageja, jotka vaikuttavat siihen, kuinka XMI esittää UML-attribuutteja skeemoissa. Näitä ovat Grosen et al. (2001) mukaan:

- xmiName
- serialize
- attribute
- element
- includeNils
- enforceMaximumMultiplicity
- enforceMinimumMultiplicity
- form
- defaultValue
- fixedValue

XMIName-tagin arvo on nimi, jota XMI käyttää elementin määrityksen nimenä ja attribuutin määrityksessä UML-attribuuteille skeemoissa.

Serialize-tagi on oletuksena asetettu todeksi, mutta on eräitä tilanteita, joissa sitä ei ole mahdollista käyttää. Esimerkiksi silloin, kun UML- attribuutin arvo voi johtua muiden attribuuttien arvoista. Johdetun attribuutin arvon voi laskea muista attribuuttiarvoista, joten niitä ei tarvitse serialisoida, kun tallennetaan objekteja. Jos attribuutti asetetaan epätodeksi, XMI ei tee XML-elementin määritystä eikä XML- attribuutin määritystä attribuutille.

Joillekin UML-attribuuteille XMI luo sekä XML-elementtimäärityksen että XML-attribuuttimäärityksen. XMI:n avulla voidaan tarvittaessa luoda ainoastaan XML- elementtimääritys UML-attribuutille. Voidaan myös antaa XMI:n luoda vain XML:n attribuuttimäärityksen UML-attribuutille. Nämä voi saada asettamalla joko elementti- tai attribuuttitagi todeksi. Jos elementtitagi on tosi, XMI luo vain XML-elementtimäärityksen ja jos attribuuttitagi on tosi, XMI luo vain XML-attribuuttimäärityksen UML- attribuutille.

XMI:ssä on tosin eräitä rajoituksia näille tageille. Molempia tageja ei voi asettaa yhtä aikaa todeksi. Attribuuttitagia ei voi asettaa todeksi attribuuteille, joilla voi olla useita arvoja. Attribuuttitagia ei voi asettaa todeksi UML-attribuuteille tyypeille, jotka ovat luokkia. Elementtitagia ei voi asettaa todeksi UML-attribuutille, jos luokassa, jossa attribuutti on, on tyhjä arvo contentType tagissa.

Jos includeNils-tagi on asetettu todeksi, XMI luo XML-elementin määritykset , jotka mahdollistavat tyhjien arvojen kirjoittamisen XMI- dokumenttiin. Tällöin XML-elementin määrityksien täytyy sisältää XML-attribuutti nillable ja se täytyy asettaa todeksi.

Oletuksena XMI luo elementtimäärityksen UML-attribuuteille ilman UML:n attribuutin multiplicity-ominaisuuden käyttöä. Mikä tahansa määrä XML-elementtejä voi esiintyä sisällössä, huolimatta UML-attribuutin kertautumisesta. XMI käyttää kertautumista UML-attribuutille, jos asetetaan arvo true enforceMinimumMultiplicity- tai enforceMaximumMultiplicity- tageille.

Voidaan määrittää, että XMI pakottaa minimimäärän, maksimimäärän tai jopa molemmat rajat. Jos näistä tageista yksi tai molemmat asetetaan todeksi, ordered-tagi luokassa, johon attribuutti kuuluu, täytyy olla asetettu todeksi.

Jos arvo asetetaan defaultValue-tagille, tätä arvoa käytetään oletusarvona XML:n attribuuttimäärityksissä UML-attribuutille. Jos fixedValue-attribuutille annetaan arvo, sitä arvoa käytetään käytettynä arvona XML-attribuuttimäärityksessä UML-attribuutille. Vain toiselle näistä tageista voidaan asettaa arvo.

2.3 SVG

2.3.1 Yleistä SVG:stä

SVG on XML-pohjainen kieli, jolla kuvataan kaksiulotteista grafiikkaa. Siinä on kaksi osaa eli XML-pohjainen tiedostoformaatti ja API-ohjelmointirajapinta graafisille sovelluksille Lilleyn ja Jacksonin (2003) mukaan. SVG tarkoittaa tarkkaan ottaen skaalautuvaa vektorigrafiikkaa englanninkielisen nimensä kirjaimien perusteella, jossa on käytetty XML-kielioppia tyylitellylle grafiikalle XML-nimiavaruudessa. Skaalautuva tarkoittaa, ettei grafiikka ole sidottu johonkin tiettyyn pikselikokoon Ferraiolon et al. (2003a) mukaan.

SVG-grafiikka on skaalautuva erilaisille näyttöresoluutioille ja sitä voidaan suurentaa tarkempiin yksityiskohtiin. SVG sallii kolmenlaisia graafisia objekteja, joita ovat vektorigrafiikkamuodot, kuvat ja tekstit. Graafiset objektit voidaan ryhmitellä, tyylittää, muuntaa ja rakentaa renderöidyistä objekteista. SVG:n ominaisuuksia ovat myös sisäiset muunnokset, leikepolut, maskaukset, filtteröinnit ja väliaikaiset objektit.

SVG:n perusmuotoja ovat neliö, ympyrä, ellipsi, viiva, murtoviiva ja polygoni. Piirtopinta eli Canvas on pinta, johon graafiset elementit piirretään ja se voi olla joko fyysinen pinta tai näyttö tai sitten jopa abstrakti pinta tietokoneen muistissa. Perusmuodot on esitetty taulukossa 3 Sallin (1999) mukaan.

Taulukko 3: SVG:n peruselementit.

Elementti	*Attribuutit*
<rect>	x,y (vasemman yläkulman koordinaatit),pituus, korkeus (sivujen pituudet);rx, ry (kulmaellipsien säteiden pituudet, jos pyöristetty).
<circle>	cx, cy (ympyrän keskipisteen koordinaatit), r (säteen pituus).
<ellipse>	cx, cy (ellipsin keskipisteen koordinaatit), rx, ry (x-akselin ja y-akselin säteiden pituudet).
<line>	x1, y1, x2, y2 (viivan päätepisteiden koordinaatit).
<polyline>	pisteet (lista pisteiden koordinaatteja).
<polygon>	pisteet (lista pisteiden koordinaatteja).

MIME-tyyppi SVG:lle on imagesvg+xml. On suositeltavaa, että SVG-tiedostoilla on loppuosa .svg

kaikilla alustoilla.

SVG 1.1 nimiavaruus, julkinen tunniste, järjestelmän tunniste ja tyyppimäärittely on määritelty kuvassa 14 Ferraiolon et al. (2003b) mukaan.

```
SVG Namespace:
http://www.w3.org/2000/svg

Public Identifier for SVG 1.1:
PUBLIC "-//W3C//DTD SVG 1.1//EN"

System Identifier for the SVG 1.1 Recommendation:
http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
         "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

Kuva 14: SVG 1.1 nimiavaruus, julkinen tunniste ja järjestelmän tunniste sekä tyyppimäärittely.

2.3.2 SVG:n käsitteiden määritelmiä

Tässä on kuvattu tutkielman kannalta merkityksellisiä määrityksiä Ferraiolon et al. (2003b) mukaan:

- *Perusmuoto*

  Standardit muodot, jotka on esimääritelty SVG:ssä sopivaksi tavallisimmille graafisille operaatioille. Erityisesti: rect,circle, ellipse,line, polyline, polygon.

- *Piirtopinta*

  Pinta, johon graafinen elementti piirretään, joka voi olla oikea fyysinen media kuten näyttö tai paperi tai abstrakti pinta, kuten alue tietokoneen muistissa.

- *Leikkauspolku*

  Yhdistelmä polusta, tekstistä ja perusmuodoista, jotka palvelevat hahmotelmana 1-bittisenä maskina, jossa kaikki hahmotelman sisällä on sallittu näyttää, mutta kaikki ulkopuolella on maskattu pois.

- *Säiliöelementti*

  Elementti, jolla voi olla graafisia elementtejä ja muita säiliöelementtejä lapsielementeinä. Erityisesti: svg, g, defs, marker ja a tässä tutkielmassa.

- *Täyttö*

  Maalausoperaatio, joka määrää muodon sisällön tai merkkijonon merkkiglyyfien sisällön.

- *Fontti*

  Fontti esittää glyyfikokoelmaa.

- *Glyyfi*

  Glyyfi on fontin kirjoitusmerkin esitysmuoto. Usein, on olemassa yksi yhteen vastaavuus piirrettyjen merkkien ja vastaavien glyyfien kesken (toisin sanoen, usein merkki A renderöidään käyttäen yksittäistä glyyfiä), mutta joskus useita glyyfejä käytetään muodostamaan yksittäinen merkki tai yksittäistä glyyfiä voidaan käyttää renderöimään useita merkkejä. Tyypillisesti glyyfi määritellään yhdellä tai useammalla muodolla kuten polulla, mahdollisesti lisätiedolla, kuten renderöintivinkeillä , jotka auttavat fonttimoottoria tuottamaan luettavissa olevaa pienikokoista tekstiä.

- *Graafinen elementti*

  Yksi elementtityyppi, joka voi aiheuttaa grafiikan piirtämisen kohdepiirtopinnalle. Erityisesti: path, text, rect, circle, ellipse, line, polyline, polygon, image ja use.

- *Maski*

  Säiliöelementti, joka voi sisältää graafisia elementtejä tai muita säiliöelementtejä , jotka määrittävät joukon grafiikkaa, jota käytetään puoliläpinäkyvänä maskina yhdistämään edustaobjekteja nykyiselle taustalle.

- *Esitysattribuutti*

  XML-attribuutti SVG-elementissä, joka määrittää arvon annetulle ominaisuudelle siinä elementissä.

- *Muoto*

  Graafinen elementti, joka on määritelty yhdistelmänä suoria viivoja ja kuvioita. Erityisesti: path, rect, circle, ellipse, line, polyline, polygon.

- *SVG-dokumenttifragmentti*

  XML-dokumentin alipuu, joka alkaa SVG-elementillä. SVG-dokumenttifragmentti voi olla kokonainen SVG-dokumentti tai fragmentti, joka on suljettu svg-elementtiin.

- *SVG-näkymäportti*

  Näkymäportti SVG-piirtopinnalla, joka määrittää suorakulmaisen alueen, johon SVG:n sisältö renderöidään.

- *Tekstiä sisältävä elementti*

  Yksi SVG:n elementti, joka voi määrittää merkkijonon, joka renderöidään piirtopinnalle. SVG:n tekstisisältöelementit ovat seuraavat: text, tspan, tref, textPath ja altGlyph.

- *Käyttäjän koordinaattijärjestelmä*

  Yleisesti koordinaattijärjestelmä määrittää suunnat ja etäisyydet käytössä olevassa piirtopinnassa. Käyttäjän koordinaattijärjestelmä on koordinaattijärjestelmä, joka on sillä hetkellä aktiivinen ja jota käytetään määrittelemään, kuinka koordinaatit ja etäisyydet sijaitsevat ja lasketaan käytössä olevalla piirtopinnalla.

- *Käyttäjän yksiköt*

  Koordinaatin arvo tai pituus kuvattuna käyttäjän esittämänä yksikkönä arvona tai pituutena sillä hetkellä olevassa käyttäjän koordinaattijärjestelmässä. Kymmenen yksikköä esittää pituutta kymmenen yksikköä käytössä olevassa käyttäjän koordinaattijärjestelmässä.

2.3.3 SVG:n sijainnin esitys

Tekstin sijainnin esittäminen SVG:llä on aika selvää, kaksi attribuuttia x ja y määrittää tekstin sijainnin kuten kuvassa 15 Archiniegasin (2004) mukaan.

Kuvassa 16 on kuvattu vastaava graafinen esitys kuvan 15 SVG-tekstin sijainnin esitykselle.

Tämä on vain pieni osa sijainnista. Ensimmäinen asia on, että x ja y ovat lista pisteitä, eivät välttämättä yksittäisiä arvoja. Sijainti, sitävastoin, ei ole rajoitettu vain yhteen arvoon tekstin alussa, vaan voi olla erityinen kullekin tekstin osalle, kuten on esitetty kuvassa 17 Arciniegasin (2004) mukaan. Kuvassa on määritelty teksti The Merry Wives sijoitettavaksi vähän korkeammalle koordinaatteihin, tämän jälkeen on määritelty seuraavan osan sijainti. Kuvassa 18 on kuvattu graafinen esitys kuvan 17 esimerkille.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-/W3C/DTD SVG 1.1/EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
          id="canvas"
       width="2000"
            height="2000"
         onload="init()">
 <text x="44" y="62">Hello World</text>
 </svg>
```

Kuva 15: SVG-tekstin sijainnin esitys.

.

Vastaavalla tavalla UML-luokka voitaisiin esittää kuvan 19 mukaisesti. Kuvassa 20 on graafinen esitys kuvan 19 esimerkille. Ylin nimiosasto ylläpitää luokan nimeä ja stereotyyppiä. Luokan nimi ja mahdollinen stereotyyppi sijoitetaan keskelle nimiosastoa. Stereotyypin arvo(ei esitetty kuvassa 19) suljetaan väkäsiin <<>>.

Attribuutin näkyvyys, symboli, nimi, tyyppi ja alkuarvo esitetään SVG-dokumentissa. Kukin attribuutti sijoitetaan erillisiin SVG-tekstielementteihin.

SVG:n esitys operaatio-osastolle on sama kuin attribuuttiosastolla. Kukin tekstielementti sisältää operaation näkyvyyden, nimen, listan parametrejä suluissa ja paluutyypin Adaptiven et al.(2003) mukaan.

Kirjoittamalla vasemmalta oikealle, vasemmalla ankkuroinnilla, sijoitus peruslinjaan, joka on oletuksena latinalaisen merkistön tapa, ei ole ainut olemassa oleva vaihtoehto. Glyyfien renderöintiä voi muuttaa määrittelemällä kirjoitussuunta, ankkurointi ja sijoituksen peruslinja. Nämä kolme SVG-piirrettä on kuvattu kuvien 21 ja 22 koodissa ja kuvan 23 graafisessa esityksessä.

Kuva 16: SVG-tekstin sijainnin visuaalinen esitys.

Tekstin ankkurointiominaisuutta (text-anchor) käytetään sijoittamaan merkkijonon alku-, keski- tai loppupiste määritettyyn pisteeseen.

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" width="450"
     height="450" version="1.1">

   <rect x="1" y="1" width="320"
         height="130"
         style="fill:#008683; stroke:black;"/>

<g style="font-family:Beachman Script; font-size: 36pt; fill:#D5E9D7">
      <text x="22 47" y="62 65">TheMerryWives</text>
      <text x="150" y="100">ofWindsor</text>
   </g>
 </svg>
```

Kuva 17: SVG sijainnin esitys.

Kuva 18: SVG-sijainnin esitys.

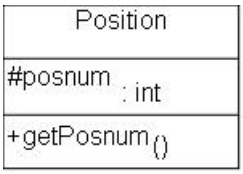
Mahdolliset arvot ominaisuudelle ovat Arciniegasin (2004) mukaan : start, middle, end, inherit.

Kirjoitustilaominaisuus (writing-mode) kontrolloi tekstin alkusuuntaa. Sitä kontrolloi kolme loogista suuntaa: vasemmalta oikealle, oikealta vasemmalle tai ylhäältä alas. Oikeat mahdolliset arvot tälle tilalle ovat: lr-tb, rl-tb, tb-rl, lr, rl, tb.

Sijainnin peruslinjaominaisuutta (alignment-baseline) käytetään määrittämään pystysuuntainen tekstin sijainti verrattuna sen vanhempaan. Mahdolliset arvot ovat: auto, baseline, before-edge, text-before-edge, middle, central, after-edge, text-after-edge, ideographic, alphabetic, hanging, mathematical. Toinen hyödyllinen ominaisuus näihin tarkoituksiin on peruslinja-siirto (baseline-shift), sen mahdolliset arvot ovat: baseline, sub, super, <percentage>, <length>, inherit.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
         "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
 <svg xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink" id="canvas"
width="2000mm" height="2000mm" onload="init()">
     <style type="text/css">
     </style>
<!--Class Position-- >
<g style="font-size:9";font-family:dialog;"
onmousemove="removePopup()"
onclick="class_click(evt,’Class Position’)"
transform="translate(184pt, 458pt)">
<g style="fill.none;stroke:black;stroke-width:1">
<rect style="fill.none;" with="99pt" height="43pt"/>
<line style="fill:none"; stroke:black;stroke-width:1" x2="99pt"
y2="20pt" y1="20pt"/>
<line style="fill:none";stroke:black;stroke-width:1 x2="99pt"
y2="41pt"
 y1="41pt"/>
</g>
<!--top name compartment-->
<text style="text-anchor">
<--attribute compartment-->
<g transform="translate(0,40)">
<text dx="2" dy="6">#<tspan x="12">posnum</tspan>:int</text>
</g>
<--operation compartment-->
<g transform="translate(0,81)">
<text class="standardfont" x="2" dy="6">+
<tspan x="12">getPosnum</tspan>()</text>
</g>
</g>
</svg>
```

Kuva 19: Luokan SVG-esitys (Gentleware AG et al., 2003).



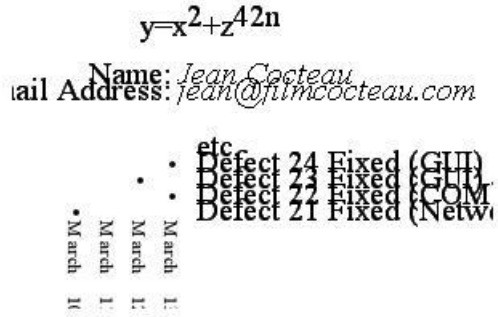
Kuva 20: Luokan SVG-esitys graafisena esityksenä (Gentleware AG et al., 2003).

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" width="450" height="450">
<!-- WRITING MODE TOP TO BOTTOM EXAMPLE -->
  <g>
  <line x1="30" y1="120" x2="30" y2="240"/>
  <text x="40" y="170" style="writing-mode:tb">March 10</text>
  <line x1="50" y1="120" x2="50" y2="240"/>
  <text x="60" y="170" style="writing-mode:tb">March 11</text>
  <line x1="70" y1="120" x2="70" y2="240"/>
  <text x="80" y="170" style="writing-mode:tb">March 12</text>
  <line x1="90" y1="120" x2="90" y2="240"/>
  <text x="100" y="170" style="writing-mode:tb">March 13</text>
  <line x1="110" y1="120" x2="110" y2="240"/>
  <line x1="20" y1="165" x2="170" y2="165"/>
  <text x="115" y="165" style="alignment-baseline:middle;
  font-size:10">Defect 21 Fixed (Networking)</text>
  <line x1="20" y1="155" x2="170" y2="155"/>
  <text x="115" y="155" style="alignment-baseline:middle;
  font-size:10">Defect 22 Fixed (COM)</text>
  <line x1="20" y1="145" x2="170" y2="145"/>
  <text x="115" y="145" style="alignment-baseline:middle;
  font-size:10">Defect 23 Fixed (GUI) </text>
  <line x1="20" y1="135" x2="170" y2="135"/>
  <text x="115" y="135" style="alignment-baseline:middle;
  font-size:10">Defect 24 Fixed (GUI) </text>
  <line x1="20" y1="125" x2="170" y2="125"/>
  <text x="115" y="125" style="alignment-baseline:middle;
  font-size:10">etc.</text>
```

Kuva 21: Perussijoitus, sijoittelu ja suunta.

```
<circle cx="40" cy="160" r="2"/>
  <circle cx="80" cy="140" r="2"/>
  <circle cx="100" cy="150" r="2"/>
  <circle cx="100" cy="130" r="2"/>
  </g>
<!-- LABELS ALIGNED TO THE LEFT EXAMPLE -->
  <g>
  <text x="100" y="80" style="text-anchor:end;
font-size:10">Name:
</text>
  <text x="103" y="80" style="font-size:10;font-style:
  italic">Jean Cocteau</text>
<text x="100" y="90" style="text-anchor:end; font-size:10">Email
  Address:</text>
  <text x="103" y="90" style="font-size:10;font-style:
  italic">jean@filmcocteau.com</text>
  </g>
<!-- SUPERSCRIPT EXAMPLE -->
  <g>
  <text x="80" y="50" style="font-size:10">y=x<tspan
  style="baseline-shift:super">2</tspan>+z
  <tspan style="baseline-shift:super">42n</tspan></text>
  </g>
  </svg>
```

Kuva 22: Perussijoittelu, sijainti ja suunta (jatkuu).



Kuva 23: Perussijoittelu, sijainti ja suunta graafisena näkymänä.

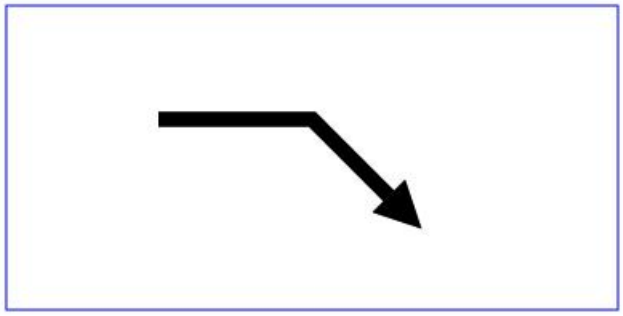
2.3.4 Merkitsimet

Merkitsin (marker) on symboli, joka on kiinnittyneenä yhteen tai useampaan path-, line-, polygon- tai polyline-elementin kärkipisteisiin (SVG working group, 2003). Tyypillisesti merkitsimiä käytetään tekemään nuolenpäitä tai monimerkitsimiä. Nuolenpäät voi määritellä kiinnittämällä merkitsin path-, line- tai polyline-elementin alku- tai loppupisteeseen.

Grafiikat merkitsimelle määritellään marker-elementillä. Monimerkitsin määritellään liittämällä merkitsin em. elementtien kaikkiin kärkipisteisiin. Tietyn marker-elementin renderöimiseksi on asetettava yksi tai useampi merkitsinominaisuus (marker, marker-start, marker-mid tai marker-end) viittaamaan annetttuun marker-elementtiin. Kuvassa 24 oleva koodi piirtää kolmionmuotoisen symbolin nuolenpäänä polun loppuun ja kuva 25 näyttää graafisen esityksen.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-/W3C/DTD SVG 1.1/EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="4in" height="2in"
viewBox="0 0 4000 2000" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<defs>
<marker id="Triangle"
viewBox="0 0 10 10" refX="0" refY="5"
markerUnits="strokeWidth"
markerWidth="4" markerHeight="3"
orient="auto">
<path d="M 0 0 L 1+ 5 L 0 10 Z" />
</marker>
</defs>
<rect x="10" y="10" width="3333" height="1980">
fill="none" stroke="blue" stroke-width="10"/>
<desc>Placing an arrowhead at the end of a path.
</desc>
<path d="M 1000 750 L 2000 750 L 2500 1250"
fill="none" stroke="black" stroke-width="100"
marker-end="url(#Triangle)"/>
</svg>
```

Kuva 24: Kolmio-merkitsin polun päässä (SVG working group, 2003).



Kuva 25: Kolmio-merkitsin polun päässä, graafinen esitys.

"Marker-start"määrittää nuolenpään polun alkuun. "Marker-end"määrittää nuolenpään polun loppuun. "Marker-mid"määrittää nuolenpään, joka piirretään muihin kärkipisteisiin.

3 Tiedon muunto

Edellä käsiteltiin kuvauskieliä ja seuraavaksi esitetään miten näitä formaatteja voidaan muuntaa eri muotoihin. Aluksi käydään läpi miten XMI voidaan muuntaa UML:ksi ja miten UML voidaan muuntaa XMI:ksi ja viimeisenä tutkitaan, miten XMI voidaan muuntaa SVG:ksi.

3.1 XMI ja UML

Koska jokainen XMI-dokumentti on laajennettu XML-dokumentti, voidaan käyttää XML-rajapintoja luomaan XMI-dokumentteja. Kaksi yleisintä XML-rajapintaa ovat Document Object Model (DOM) ja Simple API for XML (SAX) Grosen et al. (2002) mukaan.

DOM esittää XML-tiedostot solmupuina. XML-jäsentäjä mahdollistaa DOM-puun kirjoituksen XMI-tiedostoon ja DOM-puun saamisen XMI-tiedostosta. Kukin solmu DOM-puussa toteuttaa Node-rajapinnan Java-paketissa org.w3c.dom. Node-rajapinta sisältää metodit solmun tietojen saamiseksi, joita ovat tyyppi, nimi, vanhempi solmu, lapsisolmut ja sisarussolmut. Se sisältää metodit lapsisolmujen lisäykseen ja poistoon. Ylimmällä hierarkiassa on dokumenttisolmu, joka sisältää dokumenttielementin, joka on dokumentin juuren muodostava XML-elementti.

Dokumenttisolmusta luodaan muita solmutyyppejä ja se mahdollistaa elementtisolmujen saannin perustuen niiden taginimiin tai XML ID:hen. Elementtisolmu vastaa XML-elementtiä XML-dokumentissa.

Attr-solmu vastaa kutakin XML-attribuuttia. Lapsisolmut elementtisolmussa esittävät elementin sisältöä. Lapsisolmut elementissä ovat joko elementtisolmuja tai tekstisolmuja. Tekstisolmu esittää merkkidatana XML-elementin sisällön, joka ei ole mukana XML CData-osastossa. Esimerkki DOM-puuta vastaavasta XML-dokumentista on kuvassa 26.

```
<?xml version="1.0" ?>
<root a1="v1"><child>Child text</child>Root text</root>
```

Kuva 26: DOM-puuta vastaava XML-dokumentti (Grose et al., 2002).

DOM-puu voi viedä kohtuullisen paljon muistia laajoissa XML-tiedostoissa. SAX-rajapinnalla ei ole tätä rajoitusta. SAX-rajapintaa käytettäessä tietoa ei automaattisesti ladata tietorakenteeseen, vaan se

täytyy tehdä itse. Käytettäessä SAX:ia, päätetään mitä rakenteita laitetaan muistiin. Tästä johtuen sovellukset, jotka lukevat XML-tiedostoja voivat käyttää vähemmän muistia kuin DOM:ia käytettäessä.

Päärajapintoja SAX:ssa kutsutaan käsittelijöiksi ja ne koostuvat takaisinkutsumetodeista, joita XML-sarjallistaminen herättää, kun se lukee XML-tiedostoa. Tässä tutkielmassa käytetään SAX2-versiota. ContentHandler- metodi informoi XML-tiedoston alusta ja lopusta ja kun joku XML- elementti alkaa tai päättyy tai kun jokin merkkidata XML-tiedostossa luetaan. ErrorHandler-rajapinta mahdollistaa sovellusten päättää, jos virhe on tapahtunut XML-tiedostossa ja onko siitä mahdollista palautua vai ei. XML-dokumentin lukeminen SAX:lla vaatii seuraavaa:

- Toteuta yksi tai useampia käsittelijöitä.
- Rekisteröi käsittelijä oliolla, joka toteuttaa XMLReader-rajapinnan.
- Herätä sen olion parse-metodi, joka toteuttaa XMLReader-rajapinnan.

On mahdollista laajentaa DefaultHandler-luokkaa org.xml.sax.helpers-paketista. Se luokka toteuttaa kaikki ContentHandler- ja ErrorHandler-rajapintojen metodit.

Koska XMLReader on rajapinta, sillä ei ole konstruktoria. Sen sijaan käytetään staattista tehdasmetodia XMLReaderFactory.createXMLReader() palaututtamiseksi instanssin XMLReaderista. XMLReaderFactory-luokassa on kaksi tällaista metodia eli:

- public static XMLReader
  createXMLReader() throws SAXException;

- public static XMLReader
  createXMLReader(String className) throws SAXException;

Luvussa 4 kerrotaan, kuinka muunnokset voidaan toteuttaa.

3.2 XMI:n muunto SVG:ksi

XSLT-kielen syntaksi on määritelty tyylisivulla, joka on hyvin muodostettua XML:ää. Tyylisivu käytännössä sisältää elementit, jotka on määritelty XSLT:llä kuten myös elementit, joita ei ole määritelty

XSLT:llä. XSLT:llä määritellyt elementit on erotettu XML-nimiavaruuden käytöllä, johon on viitattu tämän XSLT-nimiavaruuden määrityksessä.

Termi tyylisivu heijastaa tosiasiaa, että yksi tärkein XSLT:n rooli on lisätä tyyli-informatiota XML-alkuperäisdokumentteihin, kuitenkin muuttaen niitä dokumenteiksi, jotka sisältävät XSL-muotoiltuja objekteja tai kokonaan toiseen muotoon kuten HTML:ksi, XHTML:ksi tai SVG:ksi.

Muunnos, jonka XSLT määrittää, antaa säännöt muuntoon yhdestä tai useammasta lähdepuusta yhdeksi tai useammaksi tulospuuksi. Muunnos saavutetaan joukolla sääntöjä. Tulospuun rakennuksessa, solmut lähdepuusta voidaan filtteröidä tai uudelleen järjestää ja voidaan lisätä mielivaltainen rakenne. Tämä mekanismi sallii tyylisivun olevan käypä laajalle luokalle dokumentteja, joilla on vastaavanlaiset lähtöpuurakenteet.

Yksistään toimivia XSLT-prosessoreita käytetään eniten, niitä on markkinoilla useita, mutta eivät kaikki pysty käsittelemään XSLT-tyylisivuja. XSLT-prosessoria käytetään komentoriviltä antaen sille XML-lähdedokumentin lähdedokumentin nimi, tyylisivun nimi ja minkä niminen dokumentti halutaan luoda.

Tässä on lueteltu olemassa olevia vapaan lähdekoodin XSLT-prosessoreita. Nämä kaikki ovat Java-pohjaisia Holznerin ja Holznerin (2001) mukaan.

- *SAXON:* XSLT-prosesssori, joka täysin toteuttaa XSLT 1.0 ja XPath 1.0, kuten myös muita laajennuksia näihin määrityksiin.

- *Xalan Java:* Java-toteutus W3C suosituksista XSLT:lle ja XML Path Language (XPath); Java-versio kuuluisasta Apache Xalan processorista.

- *XML jäsentäjä Javalle:*

  Oraclen XSLT processori. Tukee XSLT 1.0 suositusta, tehty käytettäväksi Javan kanssa.

- *XT:* Hyvin tunnettu toteutus Javalla XSLT-suosituksista.

Suurin osa XSLT-prosessoreista on kirjoitettu Javalla, koska Java on käyttöjärjestelmäriippumaton

ohjelmointikieli. Vaikka suurin osa näistä prosessoreista tukee standardia XSLT:tä, ne voivat antaa eri tuloksia joissain tilanteissa.

Tässä on kuvattu yksi esimerkkisivu, joka muunnetaan XSLT:llä SVG:ksi eli kuvassa 27 on kuvattu XMI-luokkakaavio Business informatics groupin (2003) mukaan.

Kuvassa 28 on kuvattu saatu SVG-tiedosto.

Liitteenä 1 on kuvattu luvun 4 muunnosvälineessä käytetty XSLT-tyylisivu.

```
<XMI xmi.version = ’1.2’ xmlns:UML = ’org.omg.xmi.namespace.UML’ >
 <XMI.header>?</XMI.header><XMI.content>
   <UML:Diagram xmi.id = ’diagramm_id’
   isVisible = ’true’ name = ’webeng_vo? zoom = ’1.0’>?
    <UML:GraphNode xmi.id = ’gn1’ isVisible = ’true’>
      <UML:GraphElement.position>
        <XMI.field>140.0</XMI.field>
        <XMI.field>130.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>100.0</XMI.field>
        <XMI.field>86.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
  <UML:Uml1SemanticModelBridge xmi.id = ’smb1’ presentation = ’’>
        <UML:Uml1SemanticModelBridge.element>
          <UML:Class xmi.idref = ’Professor’/>
        </UML:Uml1SemanticModelBridge.element>
       </UML:Uml1SemanticModelBridge>
      </UML:GraphElement.semanticModel>
   </UML:GraphNode></UML:Diagram>
  <UML:Model xmi.id = ’model_id’ name = ’vo webeng’ >
   <UML:Namespace.ownedElement>?
    <UML:Class xmi.id = ’Professor’ name = ’Professor’
    visibility = ’public’
     isSpecification = ’false’ isRoot = ’false’
     isLeaf = ’false’ isAbstract = ’false’ isActive= ’false’>
       <UML:Classifier.feature>?
       </UML:Classifier.feature>
    </UML:Class>?</UML:Namespace.ownedElement></UML:Model>
    </XMI.content></XMI>
```

Kuva 27: Luokkakaavio, josta luodaan SVG-tiedosto.

```
<svg xmlns:UML="org.omg.xmi.namespace.UML"
xmlns="http://www.w3.org/2000/svg"
width="636.6152" height="256">
<!-- Class Professor -->
<rect x="140" y="130" height="86.0"
width="100.0" fill="white" stroke="black"/>
<!--Name of Class-->
<text x="163.3235" y="144" style="font-size:11px;
font-weight:bold;">Professor</text>
<!--Name/Attribute Separator-->
<line x1="141" x2="238" y1="150" y2="150"
style="stroke:black;stroke-width=5"/>
<!-- Attribute -->
<text x="142" y="165" style="font-size:11px;">
-svnr:int</text>
?
?
<!--Association-->
<path stroke="black" fill="none" d="M240 173 L490 173 "/>
<text style="font-size:10px" x="305.3863" y="194"/>
<text style="font-size:10px" x="460.4989" y="191.5">1..*
</text>
</svg>
```

Kuva 28: Luokkakaaviosta saatu SVG-tiedosto.

4 Ohjelmointiympäristöjen tuki muunnoksille

Edellä käsiteltiin tiedon muuntoa eri formaateista toisiksi ja seuraavaksi esitetään miten se käytännössä toteutetaan esimerkkiohjelmalla. Lisäksi otetaan kantaa joidenkin työkalujen tapoihin toteuttaa asiat.

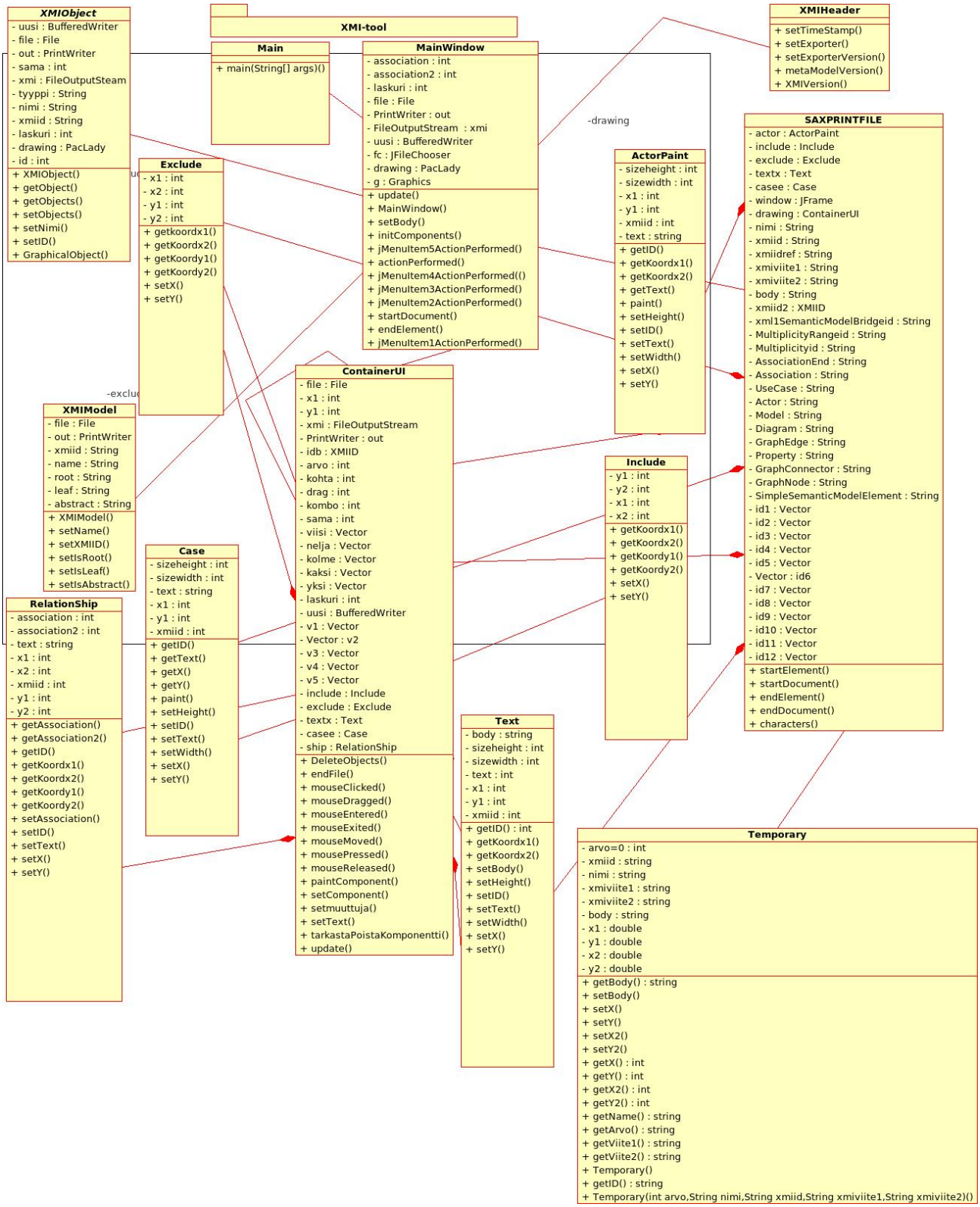
4.1 Oman muunnosvälineen rakenne

Tässä tutkielmassa käytettyjen muunnosten havainnollistamiseksi olen toteuttanut oman muuntovälineen Javalla käyttötapauskaavioiden muuntamiseksi. Toteutuksessa on hyödynnetty Swartzin (2006) ja Dacontan (2000) esittämiä ratkaisuja. Kuvassa 29 on esitelty muunnosvälineen rakenne luokkakaaviona.

Pääluokkana, jossa valikon tapahtumankäsittelijät ovat, toimii MainWindow. Näyttöluokkana toimii Container UI. Jokaisesta piirtopinnalle piirrettävästä oliosta on luotu instanssi. Näitä ovat ActorPaint toimijalle, Case käyttötapaukselle, Exclude laajennukselle, Include sisällytykselle sekä RelationShip assosiaatiolle. Luokka ContainerUI sisältää piirtopinnan tapahtumankäsittelijät ja itse piirtopinnan piirron.

Luokassa ContainerUI olevat metodit on toteutettu seuraavasti. Metodi mousePressed toimii siten, että siinä on toteutettu komponenttien liikuttaminen. Siinä tutkitaan, onko tietty komponentti hiiren klikkauspisteessä. Metodi mouseReleased on tarkoitettu sitä varten, että saadaan näkyviin dialogi assosiaatioiden nimien antamiseen ja täällä asetetaan oikeat assosiaatioiden xmi-id:t. Lisäksi tässä metodissa on toteutettu assosiaatiokomponentin poistaminen. Lisäksi täällä on myös include- ja extend-käyttötapausten luominen. Metodi setText asettaa muuttujan text nimen, jolla asetetaan komponentin nimiteksti tai kommenttiteksti. Metodi setMuuttuja asettaa muuttujan arvon, jolla tiedetään, mikä komponentti on kyseessä. Metodi setComponent asettaa komponentin id:n ja luo muut komponentit kuin assosiaation, includen ja excluden. Metodi mouseClicked tulostaa dialogit, jos halutaan muuttaa komponentteja. Lisäksi täällä on muiden kuin includen, excluden ja assosiaation poistaminen. Metodi paintComponent piirtää komponentit piirtopinnalle.

Luokassa MainWindow, joka on siis valikon tapahtumankäsittelijäluokka, metodit on toteutettu seuraavasti. jMenuItem5ActionPerformed-tapahtumankäsittelijä muuttaa SVG-tyypin XMI-muotoon. jMenuItem4ActionPerformed muuntaa XMI:n SVG:ksi. jMenuItem3ActionPerformed tallettaa XMI-muodossa tiedoston. jMenuItem2ActionPerformed avaa toisesta työkalusta tulleen XMI-tiedoston piirtopinnalle. jMenuItemActionPerformed-metodi tyhjentää piirtopinnan.



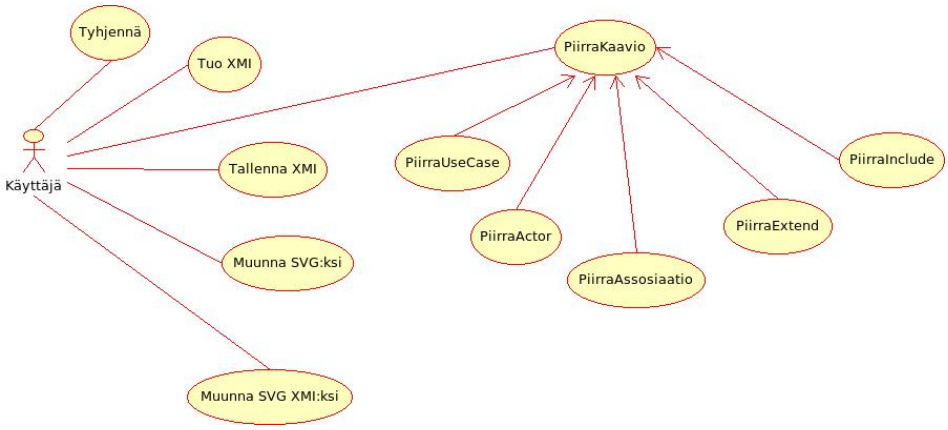
Kuva 29: Luokkakaavio muunnosvälineelle.

jMenuItem6ActionPerformed-tapahtumankäsittelijä sulkee ohjelman.

Luokka SAXPrintfile on sitä varten, että voidaan lukea XMI:tä SAX-jäsentäjällä. Luokassa on metodi startDocument, joka aloittaa dokumentin luvun. Lisäksi luokasta löytyy metodi endDocument, joka lopettaa dokumentin luvun. Joka kerran, kun luetaan uusi elementti, mennään startElement-metodiin, jonne on toteutettu sisällön lukeminen ja välittäminen muihin luokkiin. Lisäksi kunkin elementin lopettamista varten on endElement-metodi. Metodi characters tulostaa mahdolliset XMI-elementtien arvot.

Luokat XMIHeader, XMIModel, XMIObject ovat XMI:n muodostamiseen käytettäviä luokkia, XMI-Header luo XMI-otsikon siten, että se saa parametrina tiedoston, johon se kirjoittaa otsikon tiedot. Lisäksi metodi saa parametrina versionumeron, kaksi nimiavaruutta ja merkistön, XMIModel luokka luo mallin ja siinä on set-metodit määritykselle, nimelle, lehdelle, abstraktiolle ja juurelle.

XMIObject luo kunkin XMI-objektin ja tässä luokassa on metodit setxmiIdref1, setxmiIdref2, setxmiIdref3, setxmiIdref4, setx, sety, getX, getY, getKoko1, getKoko2, setKoko1, setKoko2, setNimi, setId, setIdRef. Lisäksi luokassa on parametrillinen muodostin, joka saa parametrina: isNavigable, ordering, aggregation,targetScopen, changeability, tiedosto, tyyppi, body, visibility, isSpecification, isRoot, isLeaf, isAbstract. Lisäksi luokassa on GraphicalObject-muodostin, jolla luodaan graafinen objekti ja se saa parametrina tiedoston, tyypin, koordinaatit x ja y sekä objektin koko1 ja koko2.



Kuva 30: Käyttötapauskaavio esimerkkiohjelmalle.

Kuvassa 30 on kuvattu käyttötapauskaavio muunnosvälineen toiminnoista. Käyttäjä voi valita valikosta uuden kaavion muodostamistoiminnon Tyhjennä, jolloin ohjelma tuhoaa objektit ja tyhjentää piirtopinnan. Käyttäjä voi valita ohjelman valikosta Tuo XMI, jolloin ohjelma muodostaa muunnoksella piirtopinnalle UML-kaavion. Käyttäjä voi myös muokata tuotua XMI-tietoa. Käyttäjä voi piirtää suoraan piirtopinnalle ohjelmasta löytyvillä napeilla. Käyttäjän on klikattava ensin nappia ja tä-

män jälkeen käyttäjä klikkaa piirtopintaa, minne hän haluaa objektin. Ohjelma kysyy objektin nimeä tai tekstiä. Käyttäjän painaessa ok, ohjelma tulostaa halutun olion piirtopinnalle. Käyttäjä voi poistaa olioita Delete-napilla ja siirtää olioita napin painamisen jälkeen hiirellä liikuttamalla vasen nappi pohjassa. Käyttäjä voi tallettaa piirtopinnalle luodut objektit valitsemalla valikosta Tallenna XMI. Käyttäjä voi myös muuttaa XMI- tiedoston SVG-muotoon valitsemalla valikosta Muunna SVG. Jos käyttäjä haluaa muuttaa SVG-tiedoston XMI:ksi, se onnistuu valitsemalla valikosta Muunna XMI.

Kuvan 30 käyttötapauskaaviossa oleva käyttötapaus Tallenna XMI toteutuu Liitteen 4 Java-koodissa. Tapahtumankäsittelijässä luodaan aluksi uusi.xmi -tiedosto. XMI-tiedostolla on oltava otsikko-osa, joka luodaan XMIHeader-luokalla. XMI-otsikko luodaan parametreilla uusi, joka on viite tiedostoon, version, joka on versionumero, namespace, joka on UML-nimiavaruus ja toinen UML-nimiavaruus sekä encoding, joka on merkistökoodaus. Seuraavana on vuorossa XMI -mallin luonti. Mallille annetaan parametrina luotu tiedosto. Mallille on asetettu joko oletuksena tai asettajalla, onko malli abstrakti, onko se lehti tai juuri ja sille haetaan xmi-id ja nimi. XMIObject -osuudessa luodaan eri XMI-oliot ja ne saavat muodostimessa parametrina tiedot nimestä, xmi-id:stä jne. Kullekin oliolle asetetaan oma paikkansa ja kokonsa asettimella, joka haetaan olemassa olevista olioista. XMIDiagram -osuudessa luodaan kullekin oliolle graafinen osuus ja tänne asetetaan koot ja sijainnit. Lopuksi tiedosto suljetaan.

Kuvan 30 käyttötapauskaaviossa oleva Tuo XMI- käyttötapaus toteutuu Java-koodissa liitteessä 2 olevalla tapahtumankäsittelijän koodilla, joka käyttää liitteen 3 SAXPRINTFILE-luokkaa. Aluksi luodaan Sax-jäsentäjä ja haetaan tiedosto, josta XMI-muunnetaan ruudulle UML-muotoon. Haku tapahtuu FileReader-luokalla, jolle annetaan parametrina tiedosto. Tämän jälkeen toteutetaan SAX-jäsentäjän metodit Liitteen 3 SAXPRINTFILE-luokalla. Kun luokasta on saatu luotua oliot, jotka ovat kukin omina luokkinaan, ne piirretään piirtopinnalle kutsuen ensin ActorPaint -luokassa oletuksena olevaa Paint -metodia ja tämän jälkeen kunkin kaavion elementtiluokan omaa paint-metodia.

Kuvan 30 käyttötapauskaaviossa oleva Muunna SVG:ksi-käyttötapaus toteutuu Java-koodissa liitteessä 5 olevalla tapahtumankäsittelijän koodilla. Tässä tapahtumankäsittelijässä haetaan ensin XMI-tiedosto ja tämän jälkeen muunnetaan käyttöjärjestelmän komentoriviparametrilla käyttäen xalan.jar Java-luokkaa ja xmi2svg.xsl XSLT-tyylisivua käyttäen XMI-tiedosto diagram.svg nimiseksi SVG-tiedostoksi.

Kuvan 30 käyttötapauskaaviossa oleva Muunna SVG XMI:ksi käyttötapaus toteutuu Java-koodissa liitteestä 6 löytyvällä koodilla. Seuraavassa koodissa on määritelty olioiden luonti ja muunto XMI-muotoon. Aluksi avataan SVG-tiedosto ja sieltä etsitään varattuja sanoja kuten line, ellipse ry ja path d. Kun näitä löydetään, mennään tiettyyn ehtolausehaaraan, jossa suoritetaan olion luonti. Tutkitaan, sisältääkö tietty rivi aina tiettyjä sanoja, jotka viittaavat käyttötapauksiin, toimijoihin sekä komment-

teihin. Kun oliot on löydetty ne talletetaan saaduilla arvoilla vektoreihin. Oliot käydään läpi vektoreista ja niille asetetaan arvot. Repaint metodi ContainerUI-luokassa tekee metodikutsut olioiden piirtämiseksi piirtopinnalle.

MainWindowissa XMI:n muunto SVG:ksi tapahtuu seuraavalla lauseella Liitteen 1 XSLT-tyylisivun avulla:

cmd[2] = "javaw.exe -jar xalan.jar -in "+file+-xsl xmi2svg.xsl -out diagram.svg";

XMI-tiedon talletus tapahtuu seuraavasti. Luodaan XMIHeader- ja XMIModel-oliot ja haetaan vektorista luodut oliot annetuilla arvoilla. Välissä luodaan XMIDiagram-elementti ja lopussa luodaan graafiset elementit.

4.2 Enterprise Architect-välineen XMI-tiedosto

Eri työkalut muodostavat XMI:n eri tavalla, tässä on aluksi kuvattu miten Enterprise Architect luo XMI-tiedoston. Enterprise Architect käyttää UML:Diagram nimiavaruutta kuvaamaan, missä UML-elementit näytetään ja millä tyylillä. Se myös sisältää omaa työkalukohtaista tietoa, joka voidaan sisällyttää XMI-dokumenttiin.

XMI-tiedosto jakautuu kolmeen osaan, josta ensimmäisenä on otsikko-osasto, joka on kuvattu kuvassa 31.

```
<?xml version="1.0" encoding="windows-1252">
<!DOCTYPE XMI SYSTEM "UML_EA:dtd">
<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3"
timestamp="2005-08-09 22:09:24">
<XMI.header><XMI.documentation>
<XMI.exporter>Enterprise Architect</XMI.exporter>
<XMI.exporterVersion>2.5</XMI.exporterVersion>
</XMI.documentation><XMI.header><XMI.content>
```

Kuva 31: Otsikko-osasto XMI-dokumentille.

Kuvassa 31 oleva koodi on validoitu käyttäen UML_EA_dtd:tä, kuten on merkitty DOCTYPE-tagissa, joka validoi, mitkä säännöt UML:lle ovat sallittuja. DTD-tiedosto on yksi metodi, jota XML käyttää validoidakseen sen, että XML-dokumentti on korrekti. Tämän dokumentin täytyy olla samassa hakemistossa kuin malli. Tämä malli käyttää XMI-versiota 1.1 esittämään UML-version 1.3 mallia.

UML-mallin mallielementissä sijaitsee looginen tieto. Tämä osasto sisältää tietoa, jota käytetään nimien ja attibuuttien tiedon esittämiseen mallissa. Kuvassa 32 on kuvattu malliosasto.

```
<UML:Model name="EA_Model"
<UML:Namespace.ownedElement>
<UML:Class Name="EARootClass"
xmi.id="EAID111111_5487_4080_A7F4_41526CB0AA00" isRoot="true"
isLeaf="false" isAbstract="false"/>
<UML Package name="Deployment Model"
xmi.id="EAPK_ODE50B6C_9EF5_492d_8D81_45B0B8A68599" isRoot="true"
isLeaf="false" isAbstract="false" visibility="public">
<UML:ModelElement.taggedValue>
</UML:Namespace.ownedElement>
<UML:Node name="Leasing"
xmi.id="EAID_9F5BF905_DA21_43b3_9369_1AE6E02690F0 visibility="public"
namespace="EAPK_0DE50B6C_9EF5_492d_8D81_45B0B8A68599 isRoot="false"
isLeaf="false" isAbstract="false">
<UML:ModelElement.stereotype>
<UML:Stereotype name="pc.server"/>
</UML:ModelElement.stereotype>
<UML:ModelElement.taggedValue/>
</UML:Node>
</UML:Namespace:ownedElement>
</UML:Package>
</UML:Namespace:ownedElement>
</UML:Model>
```

Kuva 32: Malliosasto UML-dokumentille.

Kuvassa 33 oleva koodi sisältää UML:Diagram elementin, jolla on attribuutteja sijainnille (missä solmu sijaitsee diagrammissa ja id sille tyylille, jota käytetään). XMI-extension tagi sisältää kullekin mallinnustyökalulle tyypillistä tietoa Goetschin (2005) mukaan. Enterprise Architect-välineellä tuotettu XMI:n versio 2.1:n ja UML:n versio 2.1:n määritysten mukainen malliosasto on kuvattu Liitteessä 7.

```
<UML:Diagram name="Deployment Model"
xmi.id="EAID_117BA063_7DBE_4850_B902_3637560C8670"
diagramType="DeploymentDiagram"
owner="EAPK_0DE50B6C_9EF5_492d_8D81_45B0B8A68599"
toolName="Enterprise Architect 2.5">
<UML:ModelElement.taggedValue/>
<UML:Diagram.element>
<UML:DiagramElement geometry="Left=315;Top=132;Right=405;Bottom=222;"
subject="EAID_9F5BF905_DA21_43b3_9369_1AE6E02690F0"
seqno="1" style="DUID=B8301E94;LBL=;"/>
</UML:Diagram.element>
</UML:Diagram>
</XMI.content>
<XMI.difference/>
<XMI.extensions xmi.extender="Enterprise Architect 2.5"/>
</XMI>
```

Kuva 33: Mallin näyttäminen Enterprise Architect-välineellä.

4.3 Mallinnustyökalujen vertailu

Seuraavaksi tarkastellaan mallinnustyökalujen Enterprise Architect, Poseidon, Together, Umbrello UML-modeler ja kohdassa 4.1 kuvatun oman muunnosvälineen UML-, XMI- ja SVG-piirteitä. Taulukossa 4 on kuvattu kullekin UML-mallinnustyökalulle ominaisia piirteitä ja sen tukemia ominaisuuksia.

Taulukko 4: Kullekin mallinnustyökalulle ominaisia piirteitä.

Väline	XMI-versio	UML-versio	XMI[DI]-tuki
Poseidon 5.01	1.0, 1.1, 1.2	2.0, 1.4	Ei
Together 6.0	1, 1.1	1.1,1.3,1.4	Ei
Oma	1.1	1.4	Kyllä
Enterprise Architect 7.0	1.0, 1.1, 1.2, 2.0, 2.1	1.1,1.2, 1.3, 2.0, 2.1	Kyllä
Umbrello UML-modeler 1.571	1.0,1.1,1.2	1.1, 1.2, 1.3, 1.4, 2.0	Ei

Taulukossa 5 on kuvattu UML-mallinnustyökalujen tiedonvaihdon testauksen onnistumista käyttötapausten osalta, koska oma työkalu mahdollistaa vain käyttötapauskaavioiden mallintamisen ja muuntamisen. Taulukon sarakkeissa on kuvattu välineet, joihin XMI-kuvaus siirretään taulukon riveillä olevista välineistä.

Taulukko 5: UML-mallinnustyökalujen tiedonvaihdon onnistuminen taulukkona.

Väline	EA	Poseidon	Together	Oma	Umbrello
EA	Kyllä	Osittain, malli näkyy	Ei tunnista UML:ää	Kyllä	Ei
Poseidon	Kyllä	Kyllä	Ei tunnista mallia	Kyllä	Malli näkyy
Together	Kyllä, malli näkyy	Ei tunnista UML:ää	Kyllä	Ei	Ei
Oma	Kyllä	Kyllä	Ei	Kyllä	Kyllä
Umbrello	Kyllä, malli näkyy	Ei	Ei	Ei	Kyllä

Togetherilla ja Umbrellolla, Poseidonilla ja omalla työkalulla kuvat voidaan tallettaa SVG-muodossa. Oma työkalu ja Umbrello UML-modeler käyttävät tallennusformaattinaan XMI:tä. "Osittain malli näkyy"taulukossa 5 tarkoittaa esimerkiksi sitä, että Enterprise Architect näyttää Poseidonissa mallin vain osittain eli siihen tulee näkyviin esimerkki mallista, muttei graafista esitystä. Togetherin kohdalla oleva "Ei tunnista UML:ää"tarkoittaa, ettei Together tunnista UML:n versiota , joka on sen hyväksymissä malleissa 2.0. Syynä siihen, että oma työkalu ottaa vastaan niin paljon erilaisia XMI-tiedostoja ja muut osaavat käsitellä niitä johtuu siitä, että tiedostoformaatti on mahdollisimman hyvin muiden työkalujen tukema ja työkalu ottaa huomioon työkalukohtaisia tietoja. Tämä näkyy myös joissain toisissa työkaluissa , kuten Poseidonin kohdalla. Umbrellon XMI-tiedostoformaatti on ainakin ennen eronnut hieman virallisesta XMI 1.1 versiosta, ehkä tässä syy sen huonoon menestykseen. Lisäksi versionumeroilla on suuri vaikutus mallin siirrettävyyteen eli mitä XMI:n ja UML:n versioita kukin ohjelma tukee, vanhemmat versiot ovat vielä paremmin tuettuja kuin uusin XMI:n versio 2.0 tai 2.1.

5 Yhteenveto

Tässä tutkielmassa käsiteltiin UML-kaavioiden muuntamista XMI-muotoon siten, että ne olisivat saatavissa UML 2.0 käsitteellisestä esityksestä ja saataisiin muunnettua helposti XMI-muotoon. Tästä ne saataisiin vielä helposti näkymään kaikille käyttäjille, joilla ei tarvitsisi välttämättä olla UML-työkalua eli tämä XMI-muoto muunnettaisiin vielä yleiseen SVG-formaattiin, joka saataisiin näkymään kunkin käyttäjän nettiselaimessa riippumatta käyttöjärjestelmästä ja laitteistosta.

UML-kaaviot voidaan muuntaa toiseen muotoon valmiilla jäsentimellä, kuten DOM, SAX tai XMI-frameworkilla, jota IBM on kehittänyt. Näiden avulla voidaan muodostaa XMI-tiedostosta olioita, jotka voidaan esittää UML-kaaviona. XMI voidaan muuntaa SVG:ksi käyttäen esimerkiksi Xercesiä ja sille täytyy antaa XSLT-tyylisivu, jolla muunto suoritetaan.

Työssä tuli selväksi, että eri UML-mallinnusohjelmien XMI-muoto vaihtelee jonkin verran, erityisesti koordinaattiesitysten ja diagrammidatan suhteen. Pääpiirteissään kaikista työkaluista saadaan pääasiallinen tieto kuitenkin ulos, vain koordinaattien ja diagrammien esitystapa vaihtelee. Tulevaisuuden haasteina voidaan nähdä se, miten UML tulee kehittymään ja pystyykö XMI-kehittymään siten, että se parantaa XMI[DI]-laajennosta siten, että siihen tulisi mukaan ulkoasun määrittelykieli.

Viitteet

Adaptive, DaimlerChrysler AG, Gentleware AG,Rational Software Corporation, Sun MicroSystems, Telelogic AB- (2003) *Unified Modeling Language: Diagram Interchange Version 2.0*. http://www.jeckle.de/files/03-07-03.pdf 5.10.2006.

Anderson, R., Birbeck, M., Kay, M., Livingstone, S., Loesgen, B., Martin, D., Mohr, S., Ozu, N., Peat, B., Pinnock, J., Stark, P., Williams, K. (2000) *Professional XML*. Prox Press Ltd., Birmingham.

Arciniegas A. (2004) *SVG and Typography*. http://www.xml.com/pub/a/2004/04//07/svgtype.html (10.5.2007)

Business informatics group (2003) *XMI+DI* http://www.big.tuwien.ac.at/teaching/offer/ss04/we_vo/weM11MDAXMI_DI.pdf (20.8.2007)

Daconta, M., C., (2000) When Runtime.exec() won't. http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html?page=3 (2.10.2007)

Elsberry,J., Elsberry,N. (2003) *Using XML and SVG to Generate Dynamic UML Diagrams* http://www.cwu.edu/~gellenbe/docs/xmltouml/xmltechnicalreport.html (6.5.2007)

Ferraiolo, J., Fujisawa, J.,Jackson,D. (2003a) *SVG specification*. http://www.w3c.org/TR/SVG/concepts.html (5.10.2006).

Ferraiolo, J., Fujisawa, J.,Jackson,D. (2003b) *SVG specification*. http://www.w3c.org/TR/2000/03/WD-SVG-20000303/intro.html (5.10.2006).

Gentleware AG, DaimlerChrysler AG, Telelogic AB, Adaptive, Rational Software Corporation, Sun Microsystem (2003) *UML 2.0 Diagram Interchange.* http://jeckle.de/files/UML2DIRevSub.pdf (5.3.2007).

Goetch, M. (2005) *XMI:Exchanging Models and Interchanging Ideas Using UML*. http://www.devx.com/enterprise/Article/28939 (10.5.2007)

Grose, T.J., Doney, G.C.,Brodsky,S.A. (2001) *Mastering XMI:Java programming with XMI,XML and UML*.. John Wiley & Sons.

H W3C XML Core Working Group (2006) Extensible Markup Language (XML) 1.0 (Fourth Edition) (2006) http://www.w3.org/TR/xml (24.2.2007).

Holzner, S, Holzner, S. (2001) em XSLT Transormations http://www.peachpit.com/articles/article.aspx?p=21845&seqNum=3&rl=1 (20.8.2007)

IBM corporation (2005) *Relationships between model elements* http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.xtools.modeler.doc/topics/cinclude.html (1.9.2007)

Koskimies,K., Koskinen,J., Maunumaa,M., Peltonen,J., Selonen,P., Siikarla,M., Systä,T. *UML työvälineenä ja tutkimuskohteena* www.tkts.fi/lehti/a21/uml.pdf (5.3.2007).

Miller,R.,(2003) *Practical UML: A Hands-On Introduction for Developers* http://dn.codegear.com/article/31863 (15.09.2007)

Lilley, C., Jackson, D. (2003) *About SVG*. http://www.w3.org/Graphics/SVG/About.html 5.10.2006.

OMG (2003) *Meta Object Facility(MOF) 2.0 XMI Mapping* www.omg.org/docs/ad/03-04-04.pdf (12.5.2007)

OMG (2007) *What is UML* http://www.omg.org/gettingstarted/what_is_uml.htm (12.5.2007)

SVG working group (2003) *Scalable Vector Graphics (SVG) 1.1 Specification*. www.w3.org/TR/SVG11/painting.html (5.10.2006).

Sall, K. (1999) *SVG Graphics Element* http://www.wdvl.com/Authoring/Languages/XML/SVG/DoingIt/graphic-elts.html (8.8.2007)

STZ-IDA and PTV AG, (2004) *xmi2svg 0.2 http://stz-ida.de/download/oss/xmi2svg.xsl (10.10.2007)*

Swartz, F.,(2006) Java: Example - Drawing a Face v2 http://www.leepoint.net/notes-java/examples/graphics/face/face2app.html (16.10.2007)

Liite 1: XSLT-tyylisivu

```
<!--

xmi2svg 0.2

Copyright (c) 2004, STZ-IDA and PTV AG, Karlsruhe, Germany

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
-->

<xsl:variable name="diagramId">
<xsl:value-of select="@xmi.id"/>
</xsl:variable>
<xsl:for-each select="//UML:Diagram[@xmi.id=$diagramId]//
UML:Property[@key='stroke' and @value != 'black' and @value != '#000000']">
<xsl:sort select="@value" data-type="text" order="ascending"/>
<xsl:variable name="index" select="position()"/>
<xsl:variable name="color">
<xsl:value-of select="@value"/>
</xsl:variable>
<xsl:variable name="omit">
<xsl:for-each select="//UML:Diagram[@xmi.id=$diagramId]
UML:Property[@key='stroke' and @value != 'black' and @value != '#000000']">
<xsl:sort select="@value" data-type="text" order="ascending"/>
<xsl:if test="position() &gt; $index and @value=$color">true</xsl:if>
<!-- note that "true" may be output a number of times,
but that doesn't matter -->
</xsl:for-each>
</xsl:variable>
<xsl:if test="$omit = ''">
<xsl:call-template name="create-markers">
<xsl:with-param name="color" select="$color"/>
</xsl:call-template>
</xsl:if>
</xsl:for-each>
```

```
</defs>
<!-- fill = #ffffe3 -->
<rect x="0" y="0" stroke="none" fill="#ffffe3" width="{$width}" height="{$height}"/>
<g>
<xsl:if test="$shiftX != 0 or $shiftY != 0">
<xsl:attribute name="transform">
<xsl:value-of select="concat('translate(', $shiftX, ',', $shiftY, ')')"/>
</xsl:attribute>
</xsl:if>
<xsl:call-template name="processGraphNode">
<xsl:with-param name="depth" select="0"/>
<xsl:with-param name="fontSize">
<xsl:call-template name="getFontSize"/>
</xsl:with-param>
<xsl:with-param name="fontColor">
<xsl:call-template name="getFontColor"/>
</xsl:with-param>
<xsl:with-param name="fontFamily">
<xsl:call-template name="getFontFamily"/>
</xsl:with-param>
</xsl:call-template>
</g>
</svg>
</xsl:template>
<!--
-  Get the marker for an AssociationEnd.
-->
<xsl:template name="getMarker">
<xsl:choose>
<xsl:when test="@aggregation='composite'">aggregation-composite</xsl:when>
<xsl:when test="@aggregation='aggregate'">aggregation-aggregate</xsl:when>
</xsl:choose>
</xsl:template>
<xsl:if test="UML:GraphElement.position">
<xsl:variable name="shiftX">
<xsl:value-of select="UML:GraphElement.position/XMI.field[1]"/>
</xsl:variable>
<xsl:variable name="shiftY">
<xsl:value-of select="UML:GraphElement.position/XMI.field[2]"/>
</xsl:variable>

<xsl:if test="$shiftX != 0 or $shiftY != 0">
<xsl:attribute name="transform">
<xsl:value-of select="concat('translate(', $shiftX, ',', $shiftY, ')')"/>
</xsl:attribute>
</xsl:if>
</xsl:if>
<xsl:when test="UML:GraphElement.semanticModel//UML:Collaboration or
                            UML:GraphElement.semanticModel//UML:UseCase">
<xsl:variable name="rx" select="UML:GraphNode.size/XMI.field[1] div 2"/>
<xsl:variable name="ry" select="UML:GraphNode.size/XMI.field[2] div 2"/>
<ellipse cx="{$rx}" cy="{$ry}" rx="{$rx}" ry="{$ry}">
<xsl:apply-templates select="." mode="style"/>
</ellipse>
```

```
</xsl:when>
<xsl:when test="UML:GraphElement.semanticModel//UML:SynchState">
<xsl:variable name="rx" select="UML:GraphNode.size/XMI.field[1] div 2"/>
<xsl:variable name="ry" select="UML:GraphNode.size/XMI.field[2] div 2"/>
<ellipse cx="{$rx}" cy="{$ry}" rx="{$rx}" ry="{$ry}">
<xsl:apply-templates select="." mode="style"/>
</ellipse>
<text x="{$rx}" y="{$ry + 4}" font-family="Helvetica" font-size="12" font-weight="bold"
text-anchor="middle">*</text>
</xsl:when>
<xsl:when test="UML:GraphElement.semanticModel// UML:Pseudostate and
key('xmi.id', UML:GraphElement.semanticModel//UML:Pseudostate/@xmi.idref)/
@kind != 'fork' and
key('xmi.id', UML:GraphElement.semanticModel//UML:Pseudostate/@xmi.idref)/
  @kind != 'join'">
<xsl:variable name="rx" select="UML:GraphNode.size/XMI.field[1] div 2"/>
<xsl:variable name="ry" select="UML:GraphNode.size/XMI.field[2] div 2"/>
<ellipse cx="{$rx}" cy="{$ry}" rx="{$rx}" ry="{$ry}">
<xsl:apply-templates select="." mode="style"/>
</ellipse>
<xsl:variable name="kind">
<xsl:value-of select="key('xmi.id', UML:GraphElement.semanticModel
                                                    UML:Pseudostate/@xmi.idref)/@kind"/>
</xsl:variable>

<!--uml actor-->
<xsl:when test="UML:GraphElement.semanticModel//UML:Actor and
not(UML:GraphElement.contained//UML:SimpleSemanticModelElement[@typeInfo='NameCompartment'])">
<xsl:variable name="width" select="UML:GraphNode.size/XMI.field[1]"/>
<xsl:variable name="height" select="UML:GraphNode.size/XMI.field[2]"/>
<line x1="0" y1="{$height}" x2="{$width div 2}" y2="{2 div 3 * $height}">
<xsl:apply-templates select="." mode="style"/>
</line>
<line x1="{$width}" y1="{$height}" x2="{$width div 2}" y2="{2 div 3 * $height}">
<xsl:apply-templates select="." mode="style"/>
</line>
<line x1="{$width div 2}" y1="{2 div 3 * $height}" x2="{$width div 2}" y2="{1.5 div 6 * $height}">
<xsl:apply-templates select="." mode="style"/>
</line>
<line x1="0" y1="{1 div 3 * $height}" x2="{$width}" y2="{1 div 3 * $height}">
<xsl:apply-templates select="." mode="style"/>
</line>
<ellipse cx="{$width div 2}" cy="{0.75 div 6 * $height}" rx="{$width * 0.175}" ry="{0.75 div 6 * $height}">
<xsl:apply-templates select="." mode="style"/>
</ellipse>
</xsl:when>
    <!-- find the association ends -->
<xsl:variable name="start-id">
<xsl:value-of select="UML:GraphElement.contained/UML:GraphNode[1]//UML:AssociationEnd/@xmi.idref"/>
</xsl:variable>
<xsl:variable name="start-marker-temp">
<xsl:for-each select="key('xmi.id', $start-id)">
<xsl:call-template name="getMarker"/>
</xsl:for-each>
```

```
</xsl:variable>
<xsl:variable name="end-id">
<xsl:value-of select="UML:GraphElement.contained/UML:GraphNode[2]//UML:AssociationEnd/@xmi.idref"/>
</xsl:variable>
<xsl:variable name="end-marker-temp">
<xsl:for-each select="key('xmi.id', $end-id)">
<xsl:call-template name="getMarker"/>
</xsl:for-each>
</xsl:variable>
<xsl:variable name="start-marker">
<xsl:choose>
<xsl:when test="$start-marker-temp = ''">
<xsl:if test="key('xmi.id', $start-id)/@isNavigable='true' and
                                          key('xmi.id', $end-id)/@isNavigable='false'">
<xsl:text>arrow</xsl:text>
</xsl:if>
</xsl:when>
<xsl:when test="$start-marker-temp = 'aggregation-composite' and
                                       key('xmi.id', $start-id)/@isNavigable='true' and
                                        key('xmi.id', $end-id)/@isNavigable='false'">
<xsl:text>aggregation-composite-arrow</xsl:text>
</xsl:when>
<xsl:when test="$start-marker-temp = 'aggregation-aggregate' and
                                     key('xmi.id', $start-id)/@isNavigable='true' and
                                        key('xmi.id', $end-id)/@isNavigable='false'">
<xsl:text>aggregation-aggregate-arrow</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$start-marker-temp"/>
</xsl:otherwise>
</xsl:choose>
</xsl:variable>
<xsl:variable name="end-marker">
<xsl:choose>
<xsl:when test="$end-marker-temp = ''">
<xsl:if test="key('xmi.id', $start-id)/@isNavigable='false' and
                                          key('xmi.id', $end-id)/@isNavigable='true'">
<xsl:text>arrow</xsl:text>
</xsl:if>
</xsl:when>
<xsl:when test="$start-marker-temp = 'aggregation-composite' and
                                           key('xmi.id', $start-id)/@isNavigable='false' and
                                          key('xmi.id', $end-id)/@isNavigable='true'">
<xsl:text>aggregation-composite-arrow</xsl:text>
</xsl:when>
<xsl:when test="$start-marker-temp = 'aggregation-aggregate' and
                                      key('xmi.id', $start-id)/@isNavigable='false' and
                                          key('xmi.id', $end-id)/@isNavigable='true'">
<xsl:text>aggregation-aggregate-arrow</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$end-marker-temp"/>
</xsl:otherwise>
</xsl:choose>
```

```
</xsl:variable>
<path d="{$commands}">
<xsl:apply-templates select="." mode="style"/>
<xsl:variable name="marker-color">
<xsl:choose>
<xsl:when test="UML:DiagramElement.property/UML:Property[@key='stroke']">
<xsl:call-template name="create-color-id">
<xsl:with-param name="color" select="UML:DiagramElement.property/UML:Property[@key='stroke']/@value"/>
</xsl:call-template>
</xsl:when>
<xsl:otherwise>
<xsl:call-template name="create-color-id">
<xsl:with-param name="color" select="'black'"/>
</xsl:call-template>
</xsl:otherwise>
</xsl:choose>
</xsl:variable>
<xsl:choose>
<xsl:when test="UML:GraphElement.semanticModel//UML:Generalization or
                                          (UML:GraphElement.semanticModel//UML:Abstraction and
                                           UML:GraphElement.contained/UML:GraphNode[@isVisible='true' and
                                          .//UML:SimpleSemanticModelElement[@typeInfo='StereotypeCompartmen
    ">
<xsl:attribute name="marker-end">url(#generalization-<xsl:value-of select="$marker-color"/>)</xsl:attribute>
</xsl:when>
<xsl:when test="UML:GraphElement.semanticModel//UML:Dependency or
                                          UML:GraphElement.semanticModel//UML:Transition or
                                     UML:GraphElement.semanticModel//UML:Include or
                                    UML:GraphElement.semanticModel//UML:Extend">
<xsl:attribute name="marker-end">url(#arrow-end-<xsl:value-of select="$marker-color"/>)</xsl:attribute>
</xsl:when>
<xsl:otherwise>
<xsl:if test="not($start-marker='arrow' and $end-marker='arrow')">
<xsl:if test="$start-marker != ''">
<xsl:attribute name="marker-start">
<xsl:text>url(#</xsl:text>
<xsl:value-of select="$start-marker"/>
<xsl:text>-start-</xsl:text>
<xsl:value-of select="$marker-color"/>
<xsl:text>)</xsl:text>
</xsl:attribute>
</xsl:if>
<xsl:if test="$end-marker != ''">
<xsl:attribute name="marker-end">
<xsl:text>url(#</xsl:text>
<xsl:value-of select="$end-marker"/>
<xsl:text>-end-</xsl:text>
<xsl:value-of select="$marker-color"/>
<xsl:text>)</xsl:text>
</xsl:attribute>
</xsl:if>
</xsl:if>
</xsl:otherwise>
</xsl:choose>
```

```
</path>
</xsl:when>

<!--uml comment->
<xsl:when test="UML:GraphElement.semanticModel//UML:Comment">
<path>
<xsl:apply-templates select="." mode="style"/>
<xsl:attribute name="d">
<xsl:text>M 0 0 L </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1] - $commentBorder"/>
<xsl:text> 0 L </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1]"/>
<xsl:text> </xsl:text>
<xsl:value-of select="$commentBorder"/>
<xsl:text> L </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1]"/>
<xsl:text> </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[2]"/>
<xsl:text> L 0 </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[2]"/>
<xsl:text> z</xsl:text>
</xsl:attribute>
</path>
<path>
<xsl:apply-templates select="." mode="style"/>
<xsl:attribute name="fill">none</xsl:attribute>
<xsl:attribute name="d">
<xsl:text>M </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1] - $commentBorder"/>
<xsl:text> 0 L </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1] - $commentBorder"/>
<xsl:text> </xsl:text>
<xsl:value-of select="$commentBorder"/>
<xsl:text> L </xsl:text>
<xsl:value-of select="UML:GraphNode.size/XMI.field[1]"/>
<xsl:text> </xsl:text>
<xsl:value-of select="$commentBorder"/>
</xsl:attribute>
</path>
</xsl:when>
<!--
-  Template for graph element position.
-->
<!--xsl:template match="UML:GraphElement.position" mode="attrib">
<xsl:attribute name="x"><xsl:value-of select="XMI.field[1]"/></xsl:attribute>
<xsl:attribute name="y"><xsl:value-of select="XMI.field[2]"/></xsl:attribute>
</xsl:template-->
<!--
-  Template for graph node size.
-->
<xsl:template match="UML:GraphNode.size" mode="attrib">
<xsl:attribute name="width"><xsl:value-of select="XMI.field[1]"/></xsl:attribute>
<xsl:attribute name="height">
<xsl:choose>
```

```
<xsl:when test="XMI.field[2]='0.0'">0</xsl:when>
<xsl:otherwise><xsl:value-of select="XMI.field[2]"/></xsl:otherwise>
</xsl:choose>
</xsl:attribute>
</xsl:template>
</xsl:stylesheet>
```

Liite 2:Tuo XMI-tapahtumankäsittelijä

```
    private void jMenuItem2ActionPerformed
(java.awt.event.ActionEvent evt) {
        int returnVal =jFileChooser1.
showOpenDialog(MainWindow.this);
        try{
            File file = jFileChooser1.getSelectedFile();
            XMLReader parser;
            try {
                parser = org.xml.sax.
helpers.XMLReaderFactory.createXMLReader(
                        "org.apache.xerces.parsers.SAXParser"
                        );
            } catch (SAXException eea) {
                try {
                parser = org.xml.sax.helpers.
XMLReaderFactory.createXMLReader();
                } catch (SAXException eeae) {
                throw new NoClassDefFoundError
("No SAX parser is available");
                }
            }
            try{
                FileReader reader=new FileReader(file);
                SAXPRINTFILE handler=new SAXPRINTFILE();
                parser.setContentHandler(handler);
                parser.setErrorHandler(handler);
                parser.parse(new InputSource(reader));
            }catch(Exception ue){
            }
        }catch(Exception ehh){}
        if(montako>0) {
            int i=0;
            if(v!=null){
                while(i<v.size()){
                    actor=(ActorPaint)v.elementAt(i);
                    if(actor!=null)
                    drawing.setComponent(xmiid, 1,actor,
casee, textx, include,exclude, ship);
                    i++;
                }
            }
            if(v1!=null) {
                i=0;
                while(i<v1.size()){
                    ship=(RelationShip)v1.elementAt(i);
                    if(ship!=null)
                    drawing.setComponent(xmiid,6,
actor, casee, textx, include,exclude, ship);
                    i++;
                }
            }
```

```
            i=0;
            if(v2!=null){
                while(i<v2.size())
                    include=(Include)v2.elementAt(i);
                    if(include!=null)
                    drawing.setComponent(xmiid, 4 ,
actor, casee, textx, include,exclude, ship);
                    i++;
                }
            }
            i=0;
            if(v3!=null){
                while(i<v3.size()){

                    exclude=(Exclude)v3.elementAt(i);
                    if(exclude!=null)
                    drawing.setComponent(xmiid,5 ,actor,
casee, textx, include,exclude, ship);
                    i++;
                }
            }
            i=0;
            if(v4!=null){
                while(i<v4.size()){
                    casee=(Case)v4.elementAt(i);
                    if(casee!=null)
                   drawing.setComponent(xmiid, 2,
actor, casee, textx, include,exclude, ship);
                    i++;
                }
            }
            i=0;
            if(v5!=null){
                while(i<v5.size()){
                   textx=(Text)v5.elementAt(i);
                    if(textx!=null){
                        drawing.setComponent(xmiid, 3 ,
actor, casee, textx, include,exclude, ship);
                    }
                    i++;
                }
            }
        }
        try {
            drawing.repaint();
        } finally {
        }
    }
```

Liite 3: SaxPrintFile-luokka

```
 /*
 * SAXPRINTFILE.java
 *
 * Created on 7. kes~kuuta 2006, 14:44
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package javaapplication9;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.Attributes;
import java.util.*;
import java.awt.*;
import javax.swing.*;
//import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.reflect.*;
import java.lang.*;
/**
 *
 * @author matti
 */
public class SAXPRINTFILE extends DefaultHandler{
    ActorPaint actor;
    Include include;
    Exclude exclude;
    Text textx;
    Case casee;
    int expo=0;
    Vector v1;
    Vector vektorit;
    JFrame window;
    ContainerUI drawing;
    String nimi="";
    String xmiid="";
    String xmiidref="";
    String xmiviite1="";
    String xmiviite2="";
    String body="";
    XMIID xmiid2;
//XMIID:t
    String xml1SemanticModelBridgeid="";
    String       MultiplicityRangeid="";
    String       Multiplicityid="";
    String AssociationEnd="";
    String    Association="";
    String         UseCase="";
    String         Actor="";
    String        Model ="";
```

```
    String         Diagram="";
    String         SimpleSemanticModelElement="";
    String         GraphNode ="";
    String         GraphConnector="";
    String         Property ="";
    String         GraphEdge="";
//vektorit id:lle
    Vector id1;
    Vector id2;
    Vector id3;
    Vector id4;
    Vector id5;
    Vector id6;
    Vector id7;
    Vector id8;
    Vector id9;
    Vector id10;
    Vector id11;
    Vector id12;
    int muuttuja=0;
    double x=100000;
    double y=0;
    double koko=100000;
    double koko2=0;
    boolean kii=false;
    boolean xy=false;
    Temporary temp;
    XMIID xmi;
    boolean xmit=false;
 /** Creates a new instance of SAXPRINTFILE */
    public SAXPRINTFILE() {
        v1=new Vector();
        id1=new Vector();;
        id2=new Vector();;
        id3=new Vector();;
        id4=new Vector();;
        id5=new Vector();;
        id6=new Vector();;
        id7=new Vector();;
        id8=new Vector();;
        id9=new Vector();;
        id10=new Vector();;
        id11=new Vector();;
        id12=new Vector();;
        xmiid2=new XMIID();
        vektorit=new Vector();
    }
    public void startDocument(){
        kii=false;
        xy=false;
    }
    public void endDocument(){}
    public void startElement(String uri, String name, String gName, Attributes atts){
        CharSequence s1="Actor";
```

```
CharSequence s2="UseCase";
CharSequence s3="Association";
CharSequence s4="xmi.id";
CharSequence s5="Comment";
CharSequence s6="Include";
CharSequence s7="Extend";
CharSequence s8="XMI";
if(name.contains(s8)){
    xmit=true;
}else {
    xmit=false;
}
if(name.contains("position")) {
    muuttuja=1;
    x=100000;
}
if(name.contains("size")){
    muuttuja=2;
}
if(name.contains("GraphEdge.waypoints")){
    muuttuja=3;
    x=100000;
    koko=100000;
}
if(atts.getLength()>0){
    String attribs="  ";
    for(int attrib=0;attrib<atts.getLength();++attrib){
        attribs=atts.getLocalName(attrib)+""+atts.getValue(attrib);
        if(name.contains("XMI.field") && attribs!=null) {
        }
        if((attribs.contains("body"))){
            body=attribs.substring(4);
            temp=new Temporary(4,nimi,xmiid,xmiviite1,xmiviite2);
            temp.setBody(body);
            v1.add(temp);
            String visibility="";
            if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
            String specification="";
            if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
            String root="";
            if(attribs.contains("isRoot")) {root=attribs.substring(16);}
            String leaf="";
            if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
            String targetScope="";
            if(attribs.contains("targetScope")) {targetScope=attribs.substring(16);}
            String changeability="";
            if(attribs.contains("changeability ")) {changeability=attribs.substring(16);}
            String abstractt="";
            if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
            String navigable="";
            if(attribs.contains("isAbstract")) {navigable=attribs.substring(16);}
            String ordering="";
            if(attribs.contains("isAbstract")) {ordering=attribs.substring(16);}
            String aggregation="";
```

```
                    if(attribs.contains("isAbstract")) {aggregation=attribs.substring(16);}
                    ParaObjekti para=new ParaObjekti("text", navigable,
ordering,aggregation, targetScope, changeability,root,leaf, abstractt,   visibility,
                            specification
                            );
                    vektorit.add(para);
                }
                if((attribs.contains("EEST"))){
                    XMIHeader header=null;
                    String time=attribs.substring(9);
                    header.getInstance().settimeStamp(time);
                }
                if((attribs.contains("version"))){
                    XMIHeader header=null;
                    String time=attribs.substring(11);
                    header.getInstance().xmiVersion(time);
                }
                if((attribs.contains("xmi.id"))&&(!attribs.contains("ref"))){
                    xmiid=attribs.substring(6);
                }
                if(name.contains("Model")){
                    XMIModel.getInstance().setXMIID(xmiid);
                        if(attribs.contains("name")){
                        String nimi=attribs.substring(4);
                        XMIModel.getInstance().setName(nimi);
                    }
                    if(attribs.contains("isSpecification")){
                        String nimi=attribs.substring(15);
                        XMIModel.getInstance().isSpecification(nimi);
                    }
                    if(attribs.contains("isRoot")){
                        String nimi=attribs.substring(6);
                        XMIModel.getInstance().setIsRoot(nimi);
                    }
                    if(attribs.contains("isLeaf")){
                        String nimi=attribs.substring(6);
                        XMIModel.getInstance().setIsLeaf(nimi);
                    }
                    if(attribs.contains("isAbstract")){
                        String nimi=attribs.substring(10);
                        XMIModel.getInstance().setIsAbstract(nimi);
                    }
                }
                else if(name.contains("Diagram")){
                    if(attribs.contains("xmi.id")){
                        String nimi=attribs.substring(4);
                        XMIDiagram.getInstance().setID(nimi);
                    }
                    if(attribs.contains("isVisible")){
                        String nimi=attribs.substring(9);
                        XMIDiagram.getInstance().setVisible(nimi);
                    }
                    if(attribs.contains("name")){
                        String nimi=attribs.substring(4);
```

```
                        XMIDiagram.getInstance().setName(nimi);
                    }
                    if(attribs.contains("zoom")){
                        String nimi=attribs.substring(4);
                        XMIDiagram.getInstance().setZoom(nimi);
                    }
                }
                if(attribs.contains("name")&& name.contains(s1)){
                    nimi=attribs.substring(4);
                    //vektorit vektoriin tiedot
                    String att="";
                    if(attribs.contains("name")){att=attribs.substring(4);}
                    String visibility="";
                    if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
                    String specification="";
                    if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
                    String root="";
                    if(attribs.contains("isRoot")) {root=attribs.substring(16);}
                    String leaf="";
                    if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
                    String abstractt="";
                    if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                    String changeability="";
                    if(attribs.contains("changeability")) changeability=attribs.substring(16);}
                    String targetscope="";
                    if(attribs.contains("targetScope")) {targetscope=attribs.substring(16);}
                    String aggregation="";
                    if(attribs.contains("aggregation")) {targetscope=attribs.substring(16);}
                    String ordering="";
                    if(attribs.contains("isAbstract")) {targetscope=attribs.substring(16);}
                    String navigable="";
                    if(attribs.contains("isAbstract")) {targetscope=attribs.substring(16);}
                    ParaObjekti para=new ParaObjekti("actor", navigable,
ordering,aggregation, targetscope, changeability,root,leaf, abstractt,visibility,
                            specification
                            );
                    vektorit.add(para);
                    temp=new Temporary(1,nimi,xmiid,xmiviite1,xmiviite2);
                    v1.add(temp);
                }
                if(attribs.contains("name")&&name.contains(s2)){
                    nimi=attribs.substring(4);
                    String visibility="";
                    if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
                    String specification="";
                    if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
                    String root="";
                    if(attribs.contains("isRoot")) {root=attribs.substring(16);}
                    String leaf="";
                    if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
                    String targetScope="";
                    if(attribs.contains("targetScope")) {targetScope=attribs.substring(16);}
                    String changeability="";
                    if(attribs.contains("changeability ")) {changeability=attribs.substring(16);}
```

```
                    String abstractt="";
                    if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                    String navigable="";
                    if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                    String ordering="";
                    if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                    String aggregation="";
                    if(attribs.contains("isAbstract")) {aggregation=attribs.substring(16);}
                    ParaObjekti para=new ParaObjekti("case", navigable,
ordering,aggregation, targetScope, changeability,root,leaf, abstractt,   visibility,
                            specification );
                    vektorit.add(para);
                    temp=new Temporary(2,nimi,xmiid,xmiviite1,xmiviite2);
                    v1.add(temp);
                }
                if((attribs.contains("name")||attribs.contains("xmi.id"))&&
name.contains(s3)&&!name.contains("End")&&!attribs.contains("ref")){
                    if(attribs.contains("name")) nimi=attribs.substring(4);
                    else nimi=attribs.substring(6);
                    if(name.contains("xml1SemanticModelBridgeid")) xml1SemanticModelBridgeid=nimi;
                    else if(name.contains("MultiplicityRangeid")) MultiplicityRangeid=nimi;
                    else if(name.contains("Multiplicityid")) Multiplicityid=nimi;
                    else if(name.contains("AssociationEnd")) AssociationEnd=nimi;
                    else if(name.contains("Association")) Association=nimi;
                    else if(name.contains("UseCase")) UseCase=nimi;
                    else if(name.contains("Actor")) Actor=nimi;
                    else if(name.contains("SimpleSemanticModelElement")) SimpleSemanticModelElement=nimi;
                    else if(name.contains("GraphNode")) GraphNode=nimi;
                    else if(name.contains("GraphConnector")) GraphConnector=nimi;
                    else if(name.contains("Property")) Property=nimi;
                    else if(name.contains("GraphEdge")) GraphEdge=nimi;
                    else if(name.contains("Model")) Model=nimi;
                    else if(name.contains("Diagram")) Diagram=nimi;
                    xmi=new XMIID( xml1SemanticModelBridgeid,
                            MultiplicityRangeid,
                            Multiplicityid,
                            AssociationEnd,
                            Association,
                            UseCase,
                            Actor,
                            Model,
                            Diagram,
                            SimpleSemanticModelElement,
                            GraphNode,
                            GraphConnector,
                            Property,
                            GraphEdge);
                    temp=new Temporary(3,nimi,xmiid,xmiviite1,xmiviite2);
                    v1.add(temp);
                    String visibility="";
                    if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
                    String specification="";
                    if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
                    String root="";
```

```
                    if(attribs.contains("isRoot")) {root=attribs.substring(16);}
                    String leaf="";
                    if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
                    String targetScope="";
                    if(attribs.contains("targetScope")) {targetScope=attribs.substring(16);}
                    String changeability="";
                    if(attribs.contains("changeability ")) {changeability=attribs.substring(16);}
                    String abstractt="";
                    if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                    String navigable="";
                    if(attribs.contains("isAbstract")) {navigable=attribs.substring(16);}
                    String aggregation="";
                    if(attribs.contains("isAbstract")) {aggregation=attribs.substring(16);}
                    String ordering="";
                    if(attribs.contains("isAbstract")) {ordering=attribs.substring(16);}
                    String visible="";
                    if(attribs.contains("isAbstract")) {visible=attribs.substring(16);}
                    ParaObjekti para=new ParaObjekti("assosiation", navigable,
ordering,aggregation, targetScope, changeability,root,leaf, abstractt,   visible,
                            specification
                            );
                    vektorit.add(para);
                }
                if(attribs.contains("xmi.idref")&&
!name.contains("End")&&!name.contains("GraphConnector")){
                    xmiidref=attribs.substring(9);
                    if(xmiviite1!="") xmiviite2=xmiviite1;
                    else
                        xmiviite2=xmiidref;
                    if(xmiviite1!=xmiidref) {
                        xmiviite1=xmiidref;
                    } else {
                        xmiviite1=xmiidref;
                        if(attribs.contains("name")) nimi=attribs.substring(4);
                        else nimi=attribs.substring(9);
                    }
                    if((name.contains("Include"))&&attribs.contains("xmi.id")){
                        if(attribs.contains("name")) nimi=attribs.substring(4);
                        else nimi=attribs.substring(6);
                        nimi=attribs.substring(4);
                        temp=new Temporary(5,nimi,xmiviite1,xmiviite2,xmiid);
                        v1.add(temp);
                        String visibility="";
                        if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
                        String specification="";
                        if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
                        String root="";
                        if(attribs.contains("isRoot")) {root=attribs.substring(16);}
                        String leaf="";
                        if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
                        String targetScope="";
                        if(attribs.contains("targetScope")) {targetScope=attribs.substring(16);}
                        String changeability="";
                        if(attribs.contains("changeability ")) {changeability=attribs.substring(16);}
```

```
                        String abstractt="";
                        if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                        String navigable="";
                        if(attribs.contains("isAbstract")) {navigable=attribs.substring(16);}
                        String ordering="";
                        if(attribs.contains("isAbstract")) {ordering=attribs.substring(16);}
                        String aggregation="";
                        if(attribs.contains("isAbstract")) {aggregation=attribs.substring(16);}
                        ParaObjekti para=new ParaObjekti("include", navigable,
ordering,aggregation, targetScope, changeability,root,leaf, abstractt,   visibility,
                                specification
                                );
                        vektorit.add(para);
                    }
                    if((attribs.contains("xmi.id")&&name.contains(s7))){
                        if(attribs.contains("name")) nimi=attribs.substring(4);
                        else nimi=attribs.substring(6);
                        nimi=attribs.substring(4);
                        temp=new Temporary(6,nimi,xmiviite1,xmiviite2,xmiid);
                        v1.add(temp);
                        String visibility="";
                        if(attribs.contains("visibility")) {visibility=attribs.substring(10);}
                        String specification="";
                        if(attribs.contains("isSpecification")) {specification=attribs.substring(16);}
                        String root="";
                        if(attribs.contains("isRoot")) {root=attribs.substring(16);}
                        String leaf="";
                        if(attribs.contains("isLeaf")) {leaf=attribs.substring(16);}
                        String targetScope="";
                        if(attribs.contains("targetScope")) {targetScope=attribs.substring(16);}
                        String changeability="";
                        if(attribs.contains("changeability ")) {changeability=attribs.substring(16);}
                        String abstractt="";
                        if(attribs.contains("isAbstract")) {abstractt=attribs.substring(16);}
                        String ordering="";
                        if(attribs.contains("isAbstract")) {ordering=attribs.substring(16);}
                        String navigable="";
                        if(attribs.contains("isAbstract")) {navigable=attribs.substring(16);}
                        String aggregation="";
                        if(attribs.contains("isAbstract")) {aggregation=attribs.substring(16);}
                        String visible="";
                        if(attribs.contains("visible")) {visible=attribs.substring(16);}
ParaObjekti para=new ParaObjekti("exclude", navigable, ordering,aggregation,
targetScope, changeability,root,leaf, abstractt,   visible,specification);
                        vektorit.add(para);
                    }
          if(muuttuja==2&&koko!=-1||name.contains("Association")||name.contains(s7)||name.contains(s6)){
              if(v1.isEmpty()!=true) {
                  int i=0;
                  while(i<v1.size()) {temp=(Temporary)v1.elementAt(i);
 if(temp!=null){
if(temp.getArvo()==1){
  if(temp.getID().contains(xmiidref)){
   MainWindow window=new MainWindow(1,x,y,koko,koko2,temp.getName(),"",vektorit);
```

```
   v1.removeElementAt(i);
  }
 }
else if(temp.getArvo()==2){
  if(temp.getID().contains(xmiidref)){
  MainWindow window=new MainWindow(2,x,y,koko,koko2,temp.getName(),"",vektorit);
  v1.removeElementAt(i);
 }
}
else if(temp.getArvo()==3){
 if(temp.getID().contains(xmiidref)){
 MainWindow window=new MainWindow(3,x,y,koko,koko2,temp.getName(),"",vektorit);
 v1.removeElementAt(i);
 }
}
//text comment
else if(temp.getArvo()==4){
 if(temp.getID().contains(xmiidref)){
 MainWindow windows=new MainWindow(4,x,y,koko,koko2,temp.getName(),temp.getBody(),vektorit);
 v1.removeElementAt(i);
 }
}
//include
else if(temp.getArvo()==5){
 if(temp.getID()==xmiidref){
 MainWindow window=new MainWindow(5,x,y,koko,koko2,temp.getName(),"",vektorit);
 v1.removeElementAt(i);
 }
}
//exclude
else if(temp.getArvo()==6){
  if(temp.getID()==xmiidref){
  MainWindow window=new MainWindow(6,x,y,koko,koko2,temp.getName(),"",vektorit);
  v1.removeElementAt(i);
  }
 }
 i++;
                    }//while
}
                            x=100000;
                            koko=100000;
                        }
                    }
                }
            }
        } else{
        }
    }
    public void endElement(String uri, String name, String qName){
    }
    public void characters(char ch[],int start, int length){
        String chars=new String(ch,start,length);
        if(chars.contains(" ")){
            if((chars.contains("Netbeans XMI Writer")||
```

```
chars.contains("Enterprise Architect"))&&(!(chars.contains("0")&&chars.contains("1")&&
chars.contains("2")&&chars.contains("3")&&chars.contains("4")&&chars.contains("5")&&
chars.contains("6")&&chars.contains("7")&&chars.contains("8")&&chars.contains("9") )))
            {
            }
        }
        if(chars.contains("0")||chars.contains("1")||chars.contains("2")||
chars.contains("3")||chars.contains("4")||chars.contains("5")||
chars.contains("6")||chars.contains("7")||chars.contains("8")||
chars.contains("9")&&xmit==true&&(!chars.contains("Netbeans XMI Writer")||
chars.contains("Enterprise Architect"))){
            if(chars.trim()!=" "){
                if(expo==0){
                    expo++;
                } else{
                }
            }
        }
        if(!chars.contains(" ")){
            if(muuttuja==1) {
                if(chars.contains("1")||chars.contains("2")||
chars.contains("3")||chars.contains("4")||chars.contains("5")||
chars.contains("6")||chars.contains("7")||chars.contains("8")||chars.contains("9")) kii=true;
                else
                    kii=false;
                if(x==100000&&kii==true) { x=Double.parseDouble(chars);
                } else {
                    y=Double.parseDouble(chars);
                    koko=100000;
                }
            }
            if(muuttuja==2) {
                if(koko==100000) { koko=Double.parseDouble(chars);
                } else{ koko2=Double.parseDouble(chars);
                }
            }
            if(muuttuja==3){
                if(chars.contains("1")||chars.contains("2")||chars.contains("3")||
chars.contains("4")||chars.contains("5")||chars.contains("6")||chars.contains("7")||
chars.contains("8")||chars.contains("9"))
                    kii=true;
                else
                    kii=false;
                if(x==100000&&kii==true) {
                    x=Double.parseDouble(chars);

                } else if(x!=100000&&kii==true&&koko==100000){
                    y=Double.parseDouble(chars);
                    koko=100001;
                }
                else  if(koko==100001&&kii==true) {
                    koko=Double.parseDouble(chars);
                    xy=true;
                } else if(koko!=100001&&kii==true&&xy==true){
```

```
                    koko2=Double.parseDouble(chars);
                    xy=false;
                }
            }
        }
    }
}
```

Liite 4:Tallenna XMI-tapahtumankäsittelijä

```
    private void jMenuItem3ActionPerformed
    (java.awt.event.ActionEvent evt) {
        try{
            xmi=new FileOutputStream(new File("uusi.xmi"));
            uusi=new BufferedWriter(new OutputStreamWriter(xmi));
            String version="1.2";
            String namespace="org.omg.xmi.namespace.UML";
            String namespace2="org.omg.xmi.namespace.UML2";
            String encoding="UTF-8";
            XMIHeader header=new XMIHeader
(uusi,version,namespace,namespace2,encoding);
            XMIModel model=new XMIModel(uusi);
            model.setIsAbstract("");
            model.setIsLeaf("");
            model.setIsRoot("");
            model.setName("");
            model.setXMIID("");
            Vector v=drawing.getVector(1);
            Vector v1=drawing.getVector(2);
            Vector v2=drawing.getVector(3);
            Vector v3=drawing.getVector(4);
            Vector v4=drawing.getVector(5);
            Vector v5=drawing.getVector(6);
            Vector vali=new Vector();
            int i=0;
            if(v1!=null){
                while(i<v1.size()){
                    actor =(ActorPaint)v1.elementAt(i);
                    if(actor!=null){
XMIObject o=new XMIObject(actor.getNavigable(),actor.getOrdering(),
actor.getAggregation(),actor.getTargetScope(),actor.getChangeability(),
uusi,"1",actor.getText(),"","",actor.getRoot(),
actor.getLeaf(),actor.getAbstract());
                        o.setX(actor.getKoordx1());
                        o.setY(actor.getKoordx2());
                        o.setKoko1(actor.getKoko1());
                        o.setKoko2(actor.getKoko2());
                        vali.add(o);
                    }
                    i++;
                }
            }
            i=0;
            if(v2!=null){
                while(i<v2.size()){
                    include=(Include)v2.elementAt(i);
                    if(include!=null){
XMIObject oo=new XMIObject("","","","","",
uusi,"5","include",include.getGraphicVisible(),
include.getSpecification(),"","","");
                        vali.add(oo);
```

```
                    }
                    i++;
                }
            }
            i=0;
            if(v3!=null){
                while(i<v3.size()){
                    exclude=(Exclude)v3.elementAt(i);
                    if(exclude!=null){
                       XMIObject ooo=new XMIObject("","","","","",
uusi,"6","exclude",exclude.getGraphicVisible(),
exclude.getSpecification(),"","","");
                        vali.add(ooo);
                    }
                    i++;
                }
            }
            i=0;
            if(v4!=null){
                while(i<v4.size()){
                    casee=(Case)v4.elementAt(i);
                    if(casee!=null){
                        XMIObject oooo=new XMIObject
(casee.getNavigable(),casee.getOrdering(),
casee.getAggregation(),casee.getTargetScope(),
casee.getChangeability(),uusi,"2",
casee.getText(),casee.getVisible(),casee.getSpecification(),
casee.getRoot(),casee.getLeaf(),casee.getAbstract());
                        oooo.setX(casee.getX());
                        oooo.setY(casee.getY());
                        oooo.setKoko1(casee.getWidth());
                        oooo.setKoko2(casee.getWidth());
                        vali.add(oooo);
                    }
                    i++;
                }
            }
            i=0;
            if(v5!=null){                while(i<v5.size()){
                    textx=(Text)v5.elementAt(i);
                    if(textx!=null) {
                        XMIObject ooooo=new XMIObject(textx.getNavigable(),
textx.getOrdering(),textx.getAggregation(),
textx.getTargetScope(),textx.getChangeability(),
uusi,"3",textx.getBody(),textx.getVisible(),
textx.getSpecification(),textx.getRoot(),textx.getLeaf(),textx.getAbstract());
                        vali.add(ooooo);
                    }
                    i++;
                }
            }
            if(v!=null) {
                i=0;
                while(i<v.size()){
```

```
                    ship=(RelationShip)v.elementAt(i);
                    if(ship!=null){
XMIObject oooooo=new XMIObject(ship.getNavigable(),
ship.getOrdering(),ship.getAggregation(),
ship.getTargetScope(),ship.getChangeability(),
uusi,"4","4",ship.getVisible(),ship.getSpecification(),
ship.getRoot(),ship.getLeaf(),ship.getAbstract());
                        vali.add(oooooo);
                    }
                    i++;
                }
            }
            XMIDiagram diagram=new XMIDiagram(uusi);
            int ks=0;
            if(vali!=null){
            while(ks<vali.size()){
               XMIObject object=(XMIObject) vali.elementAt(ks);
                if(object.getTyyppi().equals("1"))
                 object.GraphicalObject(uusi,"1", object.getX(),
object.getY(),object.getKoko1(),object.getKoko2());
                  else if(object.getTyyppi().equals("2"))
                     object.GraphicalObject(uusi,"2",
object.getX(),object.getY(),object.getKoko1(),object.getKoko2());
                  else if(object.getTyyppi().equals("3"))
                   object.GraphicalObject(uusi,"3", object.getX(),
object.getY(),object.getKoko1(),object.getKoko2());
                ks++;
                }
            }
         diagram.setXMIDiagramOwner(uusi);
            uusi.close();
        }catch(Exception e){}
    }
    private void jMenuItem2ActionPerformed
(java.awt.event.ActionEvent evt) {
        int returnVal =jFileChooser1.
showOpenDialog(MainWindow.this);
        try{
            File file = jFileChooser1.getSelectedFile();
            XMLReader parser;
            try {
                parser = org.xml.sax.helpers.
XMLReaderFactory.createXMLReader(
                        "org.apache.xerces.parsers.SAXParser"
                        );
            } catch (SAXException eea) {
                try {
                parser = org.xml.sax.helpers.
XMLReaderFactory.createXMLReader();
                } catch (SAXException eeae) {
                throw new NoClassDefFoundError
("No SAX parser is available");
                }
            }
```

```
    try{
        FileReader reader=new FileReader(file);
        SAXPRINTFILE handler=new SAXPRINTFILE();
        //implement that interface
        //xml parsinta kuvaksi
       parser.setContentHandler(handler);
        parser.setErrorHandler(handler);
        parser.parse(new InputSource(reader));
    }catch(Exception ue){
    }
}catch(Exception ehh){}
if(montako>0) {
    int i=0;
    if(v!=null){
        while(i<v.size()){
            actor=(ActorPaint)v.elementAt(i);
            if(actor!=null) drawing.setComponent(xmiid, 1,actor, casee, textx, include,exclude, ship);
            i++;
        }
    }
    if(v1!=null) {
        i=0;
        while(i<v1.size()){
            ship=(RelationShip)v1.elementAt(i);
            if(ship!=null) drawing.setComponent(xmiid,6, actor, casee, textx, include,exclude, ship);
            i++;
        }
    }
    i=0;
    if(v2!=null){
        while(i<v2.size()){
            include=(Include)v2.elementAt(i);
            if(include!=null) drawing.setComponent(xmiid, 4 ,actor, casee, textx, include,exclude, shi
            i++;
        }
    }
    i=0;
    if(v3!=null){
        while(i<v3.size()){
            exclude=(Exclude)v3.elementAt(i);
            if(exclude!=null) drawing.setComponent(xmiid,5 ,actor, casee, textx, include,exclude, ship
            i++;
        }
    }
    i=0;
    if(v4!=null){
        while(i<v4.size()){
            casee=(Case)v4.elementAt(i);
            if(casee!=null) drawing.setComponent(xmiid, 2, actor, casee, textx, include,exclude, ship)
            i++;
        }
    }
    i=0;
    if(v5!=null){
```

```
            while(i<v5.size()){
                textx=(Text)v5.elementAt(i);
                if(textx!=null){drawing.setComponent(xmiid, 3,actor, casee, textx, include,exclude, ship);
                }
                i++;
            }
        }
    }
    try {
        drawing.repaint();
    } finally {
    }
}
```

Liite 5:Muunna XMI SVG:ksi-tapahtumankäsittelijä

```
  public void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        MainWindow MyCa=new MainWindow();
        BufferedWriter uusi=new BufferedWriter(new OutputStreamWriter(xmi));
        int returnVal =jFileChooser1.showOpenDialog(MainWindow.this);
        try{
            File file = jFileChooser1.getSelectedFile();
            if(file!=null){
                String osName = System.getProperty("os.name" );
                String[] cmd = new String[3];
                if( osName.equals( "Windows NT" ) ) {
                    cmd[0] = "cmd.exe" ;
                    cmd[1] = "/C" ;
                    cmd[2] = "dir";
                } else if( osName.equals( "Windows 95" ) ) {
                    cmd[0] = "command.com" ;
                    cmd[1] = "/C";
                    cmd[2] = "dir";
                } else{
                    cmd[0] = "cmd.exe" ;
                    cmd[1] = "/C" ;
                    cmd[2] = "javaw.exe -jar xalan.jar -in
"+file+" -xsl xmi2svg.xsl -out diagram.svg";
                }
                Runtime rt = Runtime.getRuntime();
                System.out.println
("Execing " + cmd[0] + " " + cmd[1]
                        + " " + cmd[2]);
                Process proc = rt.exec(cmd);
                StreamGobbler errorGobbler = new
                        StreamGobbler(proc.
getErrorStream(), "ERROR");
                StreamGobbler outputGobbler = new
                        StreamGobbler
(proc.getInputStream(), "OUTPUT");
                errorGobbler.start();
                outputGobbler.start();
                int exitVal = proc.waitFor();
                System.out.println("ExitValue: " + exitVal);
            }//jos filua ei ole
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
```

Liite 6:Muunna SVG XMI:ksi-tapahtumankäsittelijä

```
    private void jMenuItem5ActionPerformed
   (java.awt.event.ActionEvent evt) {
        int returnVal =jFileChooser1.showOpenDialog
(MainWindow.this);
        try{
            File file = jFileChooser1.getSelectedFile();
            FileInputStream virta=new FileInputStream(file);
            BufferedReader syote=new BufferedReader
            (new InputStreamReader(virta));
            String rivi=null;
            rivi=syote.readLine();
            //Actor
            CharSequence s1="line";
            //use case
            CharSequence s2="ellipse ry";
            CharSequence h1="25";
            CharSequence s3="";
            CharSequence s4="path d";
            while(rivi!=null) {
                rivi=syote.readLine();
                if(rivi!=null) {
                    if(rivi.contains("g transform")){

                        String taulu[]=new String[10];
                        int muuttuja=0;
                        char joku='(';
                        int nro=rivi.indexOf(joku);
                        int oikeanro=nro+1;
                        char joku2=')';
                        int nro2=rivi.lastIndexOf(joku2);
                        int oikeanro2=nro2-1;
                        String substring="";
                        try{substring=rivi.substring(oikeanro, oikeanro2);
                        }catch(Exception ea){
                     }
                        if(substring!="")
                            taulu=substring.split(",");
                        int u=0;
                        while(u<taulu.length)
                          if(taulu[u].contains("0")||
taulu[u].contains("1")||
taulu[u].contains("2")||taulu[u].contains("3")||
taulu[u].contains("4")||taulu[u].contains("5")||
taulu[u].contains("6")||taulu[u].contains("7")||
taulu[u].contains("8")||taulu[u].contains("9")){
                                if(muuttuja==0) {
                                    try{xx=Double.parseDouble(taulu[u]);
                                    }catch(Exception e){}
                                }
                                else if(muuttuja==1) {
                                    try{yy=Double.parseDouble(taulu[u]);
```

```
                }catch(Exception ee){}
            }
            muuttuja++;
        }
        else{
        }
        u++;
    }
}
if(rivi.contains(s1)){
    laskuri++;
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    char joku='>';
    int nro=rivi.indexOf(joku);
    int oikeanro=nro+1;
    char joku2='<';
    int nro2=rivi.lastIndexOf(joku2);
    int oikeanro2=nro2;
    try{
        String nimi=rivi.substring(oikeanro, oikeanro2);
        temp=new Temporary(1,nimi,"","","");
        vektori.add(temp);
        arvox1=122;
        arvox2=142;
        actor=new ActorPaint((int)xx,(int)yy,arvox1,arvox2,nimi);
        v.add(actor);
        }catch(Exception e){
    }
                        }
if(rivi.contains(s2)&&(rivi.contains(h1))) {
    laskuri++;
    char joku222='"';
    int nro222=rivi.indexOf(joku222);
    char joku2222='c';
    int nro2222=rivi.indexOf(joku2222);
    int oikeanro2222=nro2222;
    String nimi2222=rivi.substring(oikeanro2222,nro2222);
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    rivi=syote.readLine();
    char joku='>';
    int nro=rivi.indexOf(joku);
    int oikeanro=nro+1;
    char joku2='<';
    int nro2=rivi.lastIndexOf(joku2);
```

```
            int oikeanro2=nro2;
            String nimi=rivi.substring(oikeanro,oikeanro2);
            temp=new Temporary(2,nimi,"","","");
            vektori.add(temp);
            arvox1=101;
            arvox2=50;
            casee=new Case((int)xx,(int)yy,arvox1,arvox2,nimi);
            v4.add(casee);
        }
        if(rivi.contains(s4)&&rivi.contains("stroke")
        &&rivi.contains("fill")) {
            laskuri++;
            String taulu[]=new String[10];
            double x=0;
            double y=0;
            double x2=0;
            double y2=0;
            int muuttuja=0;
            taulu=rivi.split(" ");
            int u=0;
            while(u<taulu.length) {
                if(taulu[u].contains("0")||
           taulu[u].contains("1")||
           taulu[u].contains("2")||taulu[u].contains("3")||
           taulu[u].contains("4")||taulu[u].contains("5")||
           taulu[u].contains("6")||taulu[u].contains("7")||
           taulu[u].contains("8")||taulu[u].contains("9")){
                    if(muuttuja==0)
                        x=Double.parseDouble(taulu[u]);
                    else if(muuttuja==1) y=Double.parseDouble(taulu[u]);
                    else if(muuttuja==2) x2=Double.parseDouble(taulu[u]);
                    else if(muuttuja==3) y2=Double.parseDouble(taulu[u]);
                    muuttuja++;
                }
                else{
                }
                u++;
            }
            temp=new Temporary(1,"","","","");
            temp.setX(x);
            temp.setY(y);
            temp.setX(x2);
            temp.setX(y2);
            vektori.add(temp);
            ship=new RelationShip((int)x,(int)y,(int)x2,(int)y2);
            int ikri=0;
            String xmiids="";
            String xmiid2s="";
while(ikri<v.size()){
    actor=(ActorPaint)v1.elementAt(ikri);
    if((x>=actor.getKoordx1())&&(x<=actor.getKoordx1()+60)
    &&(y>=actor.getKoordx2())&&(y<=actor.getKoordx2()+180) ){xmiids=actor.getID();
    }
    else{
```

```
        }
        if((x2>=actor.getKoordx1())&&(x2<=actor.getKoordx1()+60)&&
        (y2>=actor.getKoordx2())&&(y2<=actor.getKoordx2()+180) ){xmiid2s=actor.getID();
        } else{
        }
        ikri++;
    }
    ikri=0;
    while(ikri<v4.size()){
        casee=(Case)v4.elementAt(ikri);

        if((x>=casee.getX())&&(x<=casee.getX()+60)&&
        (y>=casee.getY())&&(y<=casee.getY()+180) ){
            xmiids=casee.getID();
        }
        else{
        }
        if((x2>=casee.getX())&&(x2<=casee.getX()+60)&&
        (y2>=casee.getY())&&(y2<=casee.getY()+180) ){
            xmiid2s=casee.getID();
        } else{
        }
        ikri++;
    }
            ship.setAssociation(xmiids,xmiid2s);
                v1.add(ship);
            }
        }
    }
    syote.close();
    int i=0;
    if(v!=null){
        while(i<v.size()){
            actor=(ActorPaint)v.elementAt(i);
            if(actor!=null)
                drawing.setComponent(xmiid,1,
actor, casee, textx, include,exclude, ship);
            i++;
        }
    }
    if(v1!=null) {
        i=0;
        while(i<v1.size()){
            ship=(RelationShip)v1.elementAt(i);
            if(ship!=null)
                drawing.setComponent(xmiid,6,
    actor, casee, textx, include,exclude, ship);
            i++;
        }
    }
    i=0;
    if(v2!=null){
        while(i<v2.size()){include=(Include)v2.elementAt(i);
            if(include!=null) drawing.setComponent(xmiid,5,actor, casee, textx, include,exclude, ship)
```

```
                i++;
            }
        }
        i=0;
        if(v3!=null){
            while(i<v3.size()){ exclude=(Exclude)v3.elementAt(i);
                if(exclude!=null) drawing.setComponent(xmiid,6, actor,casee, textx, include,exclude, ship)
                i++;
            }
        }
        i=0;
        if(v4!=null){
            while(i<v4.size()){ casee=(Case)v4.elementAt(i);
                if(casee!=null) drawing.setComponent(xmiid,2, actor, casee, textx, include,exclude, ship);
                i++;
            }
        }
        i=0;
        if(v5!=null){
            while(i<v5.size()){ textx=(Text)v5.elementAt(i);
                if(textx!=null){drawing.setComponent(xmiid,3, actor, casee, textx, include,exclude, ship);
                }
                i++;
            }
        }
        drawing.repaint();
    }catch(IOException e){
                }
}
```

Liite 7: Malliosasto UML-dokumentille 2.1 versioiden mukaan.

```
<uml:Model xmi:type="uml:Model"
name="EA_Model" visibility="public">
<packagedElement xmi:type="uml:Package"
xmi:id="EAPK_73E1FE99_C530_4d4c_B77D_BC64A8762084"
name="Model"
visibility="public">
<packagedElement xmi:type="uml:Package"
xmi:id="EAPK_89476B3E_81F9_42a3_B1F0_86306B62F673"
name="Use Case Model"
visibility="public">
<ownedComment xmi:type="uml:Comment"
xmi:id="EAID_D4E0FB6B_FE9E_4b5c_84AE_6A0D3A7AF9F5"
body="The Use Case model is a catalogue of
system functionality described using UML Use Cases.
 Each Use Case represents a single,
repeatable interaction that a user or &amp;quot;actor&amp;quot;
experiences when using the system. &amp;#xA;&amp;#xA;
A Use Case typically includes one or more &amp;quot;scenarios&amp;quot;
which describe the interactions that go
on between the Actor and the System, and documents
 the results and exceptions that occur
from the user’s perspective.&amp;#xA;&amp;#xA;
Use Cases may include other Use Cases
as part of a larger pattern of interaction
and may also be extended by other use
cases to handle exceptional conditions"/>
<ownedComment xmi:type="uml:Comment"
xmi:id="EAID_AC3C5856_5C9D_4888_A253_E6104DD35A40"
body="Actors are the users of the
system being modeled. Each Actor will have a well-defined role,
 and in the context of that role
have useful interactions with the system.&amp;#xA;&amp;#xA;A person may perform
the role of more than one Actor,
although they will only assume one role during one use case interaction.
&amp;#xA;&amp;#xA;An Actor role may be performed
by a non-human system, such as another computer program.">
<annotatedElement xmi:idref="EAID_D6A0BFD4_E659_4330_93A2_D4D0BCB8CE7A"/>
</ownedComment>
<ownedComment xmi:type="uml:Comment"
xmi:id="EAID_7DBEBF0C_55A9_464c_A0DF_EE9D4D368031"
 body="This package contains use cases
which define how an Actor will interact with the proposed system.
 &amp;#xA;&amp;#xA;Each interaction may be
specified using scenarios, sequence diagrams, communication diagrams
and other dynamic diagrams or textual
descriptions which together how the system when viewed
 as a &amp;quot;black-box&amp;quot; interacts with a user.">
<annotatedElement
xmi:idref="EAID_10989734_ED7F_4771_9123_E54B02497A62"/>
</ownedComment>
<packagedElement xmi:type="uml:Class"
```

```
xmi:id="EAID_2463BA46_A7DD_4fef_8660_6BCB47F0455D"
 name="$help://use_case_model_pattern.htm"
visibility="public"/>
<packagedElement xmi:type="uml:Class"
xmi:id="EAID_318C9450_83CC_43e2_BB41_73AA0A27FDCB"
 name="$help://usecasediagram.htm" visibility="public"/>
<packagedElement xmi:type="uml:Class"
xmi:id="EAID_BD8A172F_9E98_41d4_A199_A8F49B57F26C"
 name="$help://actor.htm" visibility="public"/>
<packagedElement xmi:type="uml:Package"
xmi:id="EAPK_D6A0BFD4_E659_4330_93A2_D4D0BCB8CE7A"
 name="Actors" visibility="public">
<packagedElement xmi:type="uml:Actor"
xmi:id="EAID_54083B0D_CCC5_42c4_8CB8_EB65C3D129E0"
 name="User" visibility="public">
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_dst3F5375_24E2_45f0_870E_921FB50DD6F3"
 visibility="public"
association="EAID_CB3F5375_24E2_45f0_870E_921FB50DD6F3" isOrdered="false"
 isDerived="false" isDerivedUnion="false" aggregation="none">
<type xmi:idref="EAID_E5DF77A3_4D16_4c0e_86D0_D0BCAD8BE8E6"/>
</ownedAttribute>
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_dst4D4AF8_A662_4953_B641_B7615F506B24"
 visibility="public"
association="EAID_EF4D4AF8_A662_4953_B641_B7615F506B24"
 isOrdered="false"
isDerived="false" isDerivedUnion="false" aggregation="none">
<type xmi:idref="EAID_E1ECCF13_1765_450b_8165_CA6E654BD7F2"/>
</ownedAttribute>
</packagedElement>
<packagedElement xmi:type="uml:Association"
xmi:id="EAID_CB3F5375_24E2_45f0_870E_921FB50DD6F3"
visibility="public">
<memberEnd
xmi:idref="EAID_dst3F5375_24E2_45f0_870E_921FB50DD6F3"/>
<memberEnd
xmi:idref="EAID_src3F5375_24E2_45f0_870E_921FB50DD6F3"/>
<ownedEnd
xmi:type="uml:Property"
xmi:id="EAID_src3F5375_24E2_45f0_870E_921FB50DD6F3"
 visibility="public"
association="EAID_CB3F5375_24E2_45f0_870E_921FB50DD6F3"
 isOrdered="false"
isDerived="false" isDerivedUnion="false" aggregation="none">
<type xmi:idref="EAID_54083B0D_CCC5_42c4_8CB8_EB65C3D129E0"/>
</ownedEnd>
</packagedElement>
<packagedElement xmi:type="uml:Association"
xmi:id="EAID_EF4D4AF8_A662_4953_B641_B7615F506B24"
visibility="public">
<memberEnd
xmi:idref="EAID_dst4D4AF8_A662_4953_B641_B7615F506B24"/>
<memberEnd
```

```
xmi:idref="EAID_src4D4AF8_A662_4953_B641_B7615F506B24"/>
<ownedEnd
xmi:type="uml:Property" xmi:id="EAID_src4D4AF8_A662_4953_B641_B7615F506B24"
 visibility="public" association="EAID_EF4D4AF8_A662_4953_B641_B7615F506B24"
 isOrdered="false" isDerived="false" isDerivedUnion="false" aggregation="none">
<type xmi:idref="EAID_54083B0D_CCC5_42c4_8CB8_EB65C3D129E0"/>
</ownedEnd>
</packagedElement>
</packagedElement>
<packagedElement xmi:type="uml:Package"
xmi:id="EAPK_10989734_ED7F_4771_9123_E54B02497A62"
 name="Primary Use Cases" visibility="public">
<ownedComment xmi:type="uml:Comment"
xmi:id="EAID_D5087522_05B6_4a3a_9B33_314CF4E69BD2"
 body="The System Boundary shows
the logical interface between users and the system being described.">
<annotatedElement xmi:idref="EAID_C7D2C1B3_D4F1_4e27_9F93_DA576D224BC5"/>
</ownedComment>
<packagedElement xmi:type="uml:UseCase"
xmi:id="EAID_E1ECCF13_1765_450b_8165_CA6E654BD7F2"
name="Use Case1" visibility="public">
<nestedClassifier xmi:type="uml:Collaboration"
xmi:id="EAID_CB000000_F13_1765_450b_8165_CA6E654BD7F"
name="EA_Collaboration1" visibility="public">
<ownedBehavior xmi:type="uml:Interaction"
xmi:id="EAID_IN000000_F13_1765_450b_8165_CA6E654BD7F"
name="EA_Interaction1" visibility="public">
<lifeline xmi:type="uml:Lifeline"
xmi:id="EAID_5ECDEA53_F23D_44b1_BBC1_BBB6395E86EE" name="Object1"
visibility="public"
represents="EAID_AT000000_A53_F23D_44b1_BBC1_BBB6395E86E"/>
<lifeline xmi:type="uml:Lifeline"
xmi:id="EAID_LL000000_CCC5_42c4_8CB8_EB65C3D129E0" name="User"
visibility="public"
represents="EAID_AT000000_000_CCC5_42c4_8CB8_EB65C3D129E"/>
<fragment xmi:type="uml:OccurrenceSpecification"
xmi:id="EAID_FR000000_CCC5_42c4_8CB8_EB65C3D129E0"
 covered="EAID_LL000000_CCC5_42c4_8CB8_EB65C3D129E0"/>
<fragment xmi:type="uml:OccurrenceSpecification"
xmi:id="EAID_FR000000_F23D_44b1_BBC1_BBB6395E86EE"
 covered="EAID_5ECDEA53_F23D_44b1_BBC1_BBB6395E86EE"/>
<message xmi:type="uml:Message"
xmi:id="EAID_029B711E_4848_4ee0_9E7C_EBD98F02CF3D"
messageKind="complete" messageSort="synchCall"
sendEvent="EAID_FR000000_CCC5_42c4_8CB8_EB65C3D129E0"
 receiveEvent="EAID_FR000000_F23D_44b1_BBC1_BBB6395E86EE"/>
</ownedBehavior>
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_AT000000_A53_F23D_44b1_BBC1_BBB6395E86E"/>
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_AT000000_000_CCC5_42c4_8CB8_EB65C3D129E">
<type xmi:idref="EAID_54083B0D_CCC5_42c4_8CB8_EB65C3D129E0"/>
</ownedAttribute>
</nestedClassifier>
```

```
</packagedElement>
<packagedElement xmi:type="uml:UseCase"
xmi:id="EAID_E5DF77A3_4D16_4c0e_86D0_D0BCAD8BE8E6"
name="Use Case2" visibility="public"/>
<packagedElement xmi:type="uml:Class"
xmi:id="EAID_C7D2C1B3_D4F1_4e27_9F93_DA576D224BC5"
name="System Boundary" visibility="public"/>
</packagedElement>
</packagedElement>
</packagedElement>
</uml:Model>
```