Juha Rautiainen

AVOIMEN LÄHDEKOODIN UML-MALLINNUSVÄLINEIDEN VERTAILU PIENTEN OHJELMISTOPROJEKTIEN TARPEISIIN

Tietotekniikan pro gradu -tutkielma, 15.1.2014

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Juha Rautiainen

Yhteystiedot: juhar@iki.fi

Työn nimi: Avoimen lähdekoodin UML-mallinnusvälineiden vertailu pienten ohjelmistoprojektien tarpeisiin

Title in English: Comparison of Open Source UML Modeling Tools for the Needs of Small Software Projects

Työ: Tietotekniikan pro gradu -tutkielma

Sivumäärä: 85 + 12

Suuntautumisvaihtoehto: Ohjelmistotekniikka.

Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos

Avainsanat: UML, avoin lähdekoodi, mallinnuskieli, mallinnusväline, UML-tuki, käytettävyys, elinkaarikustannukset, vertailu.

Keywords: UML, open source, modeling language, modeling tool, UML Support, usability, life time costs, comparison.

Tiivistelmä: Tutkielmassa esitetään vertailukohteita UML-mallinnusvälineiden UML-tuen, käytettävyyden ja elinkaarikustannusten arvioimiseksi. Vertailukohteita on sovellettu valittaessa kolmea avoimen lähdekoodin ja yhden suljetun lähdekoodin mallinnusvälinettä, sekä arvioitaessa niiden soveltuvuutta käytettäväksi opiskelijaprojekteissa ja mikroyrityksissä.

Abstract: The study presents points of comparison for the UML modeling tools to assess the UML support, usability and life cycle costs. The chosen points of comparison are applied to choose the three open source and one closed-source modeling tool and to specify their suitability for use in student projects and micro enterprises.

Termiluettelo

BPMN	(Business Process Model and Notation) on liiketoimintaprosessien graafiseen mallintamiseen tarkoitettu standardi kuvaustapa.
Heuristiikka	on määrätyistä (esimerkiksi käytettävyyden) periaatteista koostuva sääntölista.
Kaaviotyyppi	on mallinnuskielen osakieli.
Malli	on kohteena olevasta tietojärjestelmästä laadittu kuvaus, joka esittää oleelliset osat järjestelmästä valitun näkökulman kannalta. Malleja käytetään järjestelmän ymmärtämisen ja rajaamisen apuna, sekä määrittelyjen ja dokumenttien laatimiseen.
Mallinnuskieli	on mallien esittämiseen tarkoitettu kuvaustapa, jonka rakenne ja rakenteiden merkitys on määritelty.
Mallinnusväline	on mallien laatimiseen, käsittelyyn ja varastointiin käytettävä ohjelmisto.
OMG	(Object Management Group) on organisaatio, joka vastaa muun muassa UML:n standardoinnista.
UML	(Unified Modeling Language) on ensisijaisesti tietojärjestelmien määrittelyyn, suunnitteluun, visualisointiin ja dokumentointiin tarkoitettu graafinen mallinnuskieli, jota käytetään olioperustaisessa ohjelmistosuunnittelussa.
Vapaa lähdekoodi	tarkoittaa ohjelmistojen lisensoimista siten, että niiden lähdekoodi on vapaasti saatavilla, edelleen kehitettävissä ja levitettävissä.
XMI	on OMG:n kehittämä tiedonsiirtomuoto, joka mahdollistaa UML-mallin esittämisen XML-muodossa

Sisältö

1 Johdanto

Tietojenkäsittelyn tarpeiden kasvun myötä tietojärjestelmät ovat laajentuneet ja monimutkaistuneet. Niiden kehitystyön hallinta edellyttää työvälineitä, jotka automatisoivat ja tukevat ohjelmistokehitysprosessien toimintojen tehtäviä sekä auttavat toteutettavaan ohjelmistoon liittyvän informaation käsittelyssä.

Tutkielman lähtökohtana on tarve löytää Jyväskylän yliopiston Sovellusprojekti-opintojaksolla käytettäväksi soveltuvia ohjelmistokehitystä tukevia edullisia ohjelmistoja. Erityisesti on haluttu tietoa avoimen lähdekoodin ohjelmistoista, mutta käytettäväksi voisivat soveltua lisenssiehtojen sen salliessa myös suljetun lähdekoodin ohjelmistot, joista on saatavilla esimerkiksi maksuton kokeiluversio.

Ohjelmistokehitystä tukevia ohjelmistoja on kehitetty moniin eri tarkoituksiin, kuten ohjelmistojen suunnitteluun, versiohallintaan, testaukseen ja dokumentointiin. Tutkielman aihepiiriä on työmäärän rajaamiseksi tarkennettu siten, että siinä keskitytään tarkastelemaan UML-mallien ja -kaavioiden laatimiseen, käsittelyyn ja varastointiin soveltuvia yleiskäyttöisiä graafisia ohjelmistoja, eli UML-mallinnusvälineitä. UML:n käyttötapoja ei ole standardissa rajattu, joten sitä voidaan hyödyntää eri tavoin tarpeesta ja käyttäjistä riippuen. UML:a voidaan käyttää ohjelmistojen kuvaamisen lisäksi esimerkiksi myös jonkin tarkasteltavan aihealueen käsitteiden kuvaamiseen.

Tutkielma keskittyy UML:n käyttöön ohjelmistokehityksen tukena. Tutkielman tavoitteena on kartoittaa ja vertailla avoimen lähdekoodin UML-mallinnusvälineitä, sekä määrittää ja rajata vertailukohteet, joiden perusteella arvioidaan valittujen ohjelmistojen soveltuvuutta Sovellusprojektien ja mikroyritysten tarpeisiin. Avoimen lähdekoodin mallinnusvälineiden keskinäisen vertailun lisäksi arvioidaan niiden etuja ja puutteita suhteessa verrokkina käytettyyn suljetun lähdekoodin ohjelmistoon.

Tutkielman taustaa ja tavoitteita esitellään luvussa 2. Luku 3 käsittelee UML-mallinnuskieltä, sekä sen kehitystä ja käyttöä ohjelmistokehitysprosessissa. Luvussa 4 esitellään muutamia käytettävyyden määritelmiä ja käytettävyyden arvioinnissa käytettäviä menetelmiä. Lähteissä esitettyjä vertailukohteita ja niiden käyttöä tutkielmassa esitellään

luvussa 5. Luvussa 6 kuvataan tutkielmassa tarkastellut ohjelmistot sekä esitetään niiden arvioinnit ja johtopäätösten perusteella tehdyt suositukset.

2 Taustaa ja tutkielman tavoitteet

Tutkielman tavoitteena on määrittää avoimen lähdekoodin UML-mallinnustyökalujen arviointiin soveltuvat vertailukohteet ja arvioida niihin perustuen muutamien UML-mallinnusvälineiden soveltuvuutta käytettäväksi Sovellusprojekti-opintojaksolla sekä mahdollisesti muutaman työntekijän mikroyrityksissä.

Sovellusprojekti-opintojaksoa kuvataan luvussa 2.1 ja mikroyritysten toimintaympäristöstä tehtyjä oletuksia luvussa 2.2. Vapaan lähdekoodin määrittely on esitetty luvussa 2.3. Tutkielman tavoitteet ja tehdyt rajaukset kuvataan luvussa 2.4

2.1 Sovellusprojekti-opintojakso

Sovellusprojekti on kuvauksensa [Santanen 2008] mukaan Jyväskylän yliopiston tietotekniikan laitoksen pääaineopiskelijoiden syventäviin opintoihin kuuluva opintojakso. Sen tavoitteena on antaa tietotekniikan opiskelijoille käsitys työelämän ohjelmistoprojektien läpiviennistä ja ryhmätyöstä sekä projekteihin liittyvästä suullisesta ja kirjallisesta viestinnästä. Samalla opiskelijat saavat kokemusta kurssien harjoitustöitä laajemman ohjelmiston määrittelystä, suunnittelusta, toteuttamisesta ja testaamisesta sekä dokumentoinnista.

Kunkin sovellusprojektina kehitettävän ohjelmiston toteuttaa ryhmätyönä 3–5 opiskelijaa. Sovellusprojekteihin osallistuvilla opiskelijoilla on ollut vaihteleva määrä opintoja takanaan, mutta pääosin he ovat olleet kolmatta tai neljättä vuotta tietotekniikkaa pääaineenaan opiskelevia. Opiskelijakohtainen työpanos on 200–400 tuntia, ja projekti suoritetaan noin neljän kuukauden aikana syys- tai kevätlukukaudella. Rajatusta ajasta ja työpanoksesta johtuen tilaaja joutuu lähes poikkeuksetta jatkokehittämään projektin tuloksena syntyvää sovellusta.

Sovellusprojekti-opintojaksolla kehitettyjä ohjelmistoja on esitelty WWW-sivulla [Heikkilä]. Viiden viimeisimmän vuoden projekteissa ovat käytetyimpiä tekniikoita olleet Ruby on Rails -sovelluskehys, sekä ohjelmointikielet Ruby, JavaScript ja Python. Ryhmien jäsenillä on käytössään sekä Windows- että Linux-työasemia.

Opintojakson oppimistavoitteiden kannalta on toivottavaa, että projektiryhmä mallintaa vähintään toteutettavan sovelluksen keskeisimmät osat ja niiden vuorovaikutuksen ennen toteutuksen aloittamista. Myös kehitettävällä ohjelmistolla tuettavan prosessin määrittäminen ja kuvaaminen on usein tarpeellista. Mallintamisesta vastaa Jukka-Pekka Santasen mukaan tyypillisesti yksi ryhmän jäsenistä.

Projektiryhmä laatii Sovellusprojektien ohjeen [Santanen 2013] mukaisen dokumentaation. Sovellusraportissa tai luokkadokumentaatiossa tulee esimerkiksi kuvata sovelluksen kokonaisrakenne, jakaminen luokkiin, rajapinnat ja oleellisimmat tietorakenteet. Dokumentoinnissa käytetään luvussa 3.3 kuvatuista kaaviotyypeistä luokka-, käyttötapaus-, sekvenssi-, aktiviteetti- ja/tai tilakaaviota.

2.2 Mikroyritykset

Mikroyrityksellä tarkoitetaan Euroopan yhteisöjen komission määritelmän [Euroopan yhteisöjen komissio] mukaan pienten ja keskisuurten yritysten luokassa yritystä, jonka palveluksessa on alle 10 työntekijää ja jonka vuosiliikevaihto tai taseen loppusumma on enintään 2 miljoonaa euroa.

Tilastokeskuksen Informaatiopalvelujen tilinpäätöstilaston [Tilastokeskus] mukaan luokan *Ohjelmistot, konsultointi ja siihen liittyvä toiminta* 5204 yrityksestä 5092 kuului pienten ja keskisuurten yritysten luokkaan vuonna 2011. Pienten ja keskisuurten yritysten yhteenlaskettu liikevaihto oli 2,7 miljardia eroa ja henkilöstön yhteenlaskettu määrä 20 052 henkilöä. Näin ollen suuri osa yrityksistä lukeutuu sekä liikevaihtonsa että henkilöstömääränsä mukaan mikroyritysten luokkaan.

Vertailukohteita valittaessa voitaneen olettaa, että mikroyritysten tarpeet vastaavat UML-mallinnuksen osalta paljolti Sovellusprojekti-opintojakson tarpeita. Koska mikroyrityksissä ei todennäköisesti kokonsa vuoksi ole juuri käytettävissä resursseja uuden työkalun käyttöönoton suunnitteluun ja kouluttamiseen, ovat hankintahinnan ja käyttökustannusten ohella käyttöönoton helppous ja mallinnusvälineen opittavuus merkittäviä työkalua valittaessa.

Artikkelissa [Koskimies, s. 21] arvioidaan yritysten käyttävän UML:sta vain itselleen tarpeellisia osajoukkoja. Kirjassa [Booch, s. 7] kehotetaan valitsemaan mallin tarkkuus ja

laajuus suhteessa projektin monimutkaisuuteen. Luokkakaavio on kaaviotyypeistä tunnetuin, joten sen voidaan olettaa olevan käytetyin myös mikroyrityksissä.

2.3 Vapaan lähdekoodin määritelmä

Vapaan lähdekoodin ohjelmisto on sovellus, joka on lisensoitu vapaan lähdekoodin lisenssillä. Vapaan lähdekoodin lisensseissä edellytetään tiettyjen vapaan lähdekoodin periaatteiden noudattamista, kuten ohjelmiston lähdekoodin saattamista julkisesti saataville. Jos ohjelmisto ei ole avoimen lähdekoodin ohjelmisto, siitä käytetään nimitystä **suljetun lähdekoodin ohjelmisto**.

Vapaan lähdekoodin ohjelmistojen määrittelyssä on kaksi koulukuntaa. Näkemyseroista huolimatta määrittelyt pohjautuvat samoihin periaatteisiin. Vapaan lähdekoodin ohjelmiston määritelmä voi perustua joko Open Source Iniativen (OSI) [Open Source Iniative] tai Free Software Foundationin (FSF) [Free Software Foundation] määritelmään. OSI käyttää termiä **avoin lähdekoodi** (engl. *open source*) ja FSF termiä **vapaa ohjelmisto** (engl. *free software*). Viitattaessa yleisesti OSI:n ja FSF:n määrittelyn mukaisiin ohjelmistoihin käytetään tutkielmassa termiä vapaa lähdekoodi.

2.4 Tutkielman tavoitteet ja rajaus

Tutkielman lähtökohtana on tarve löytää Sovellusprojekti-opintojaksolla käytettäväksi soveltuvia ohjelmistokehitystä tukevia edullisia ohjelmistoja. Tavoitteena on

- määrittää ja rajata vertailukohteet käytettäväksi mallinnusvälineiden arviointiin ja vertailuun,
- kartoittaa Sovellusprojekti-opintojaksolla läpivietävän ohjelmistoprojektin kaltaisiin käyttötilanteisiin mahdollisesti soveltuvia avoimen lähdekoodin UML-mallinnusvälineitä, sekä
- arvioida ja vertailla kolmen valitun mallinnusvälineen soveltuvuutta Sovellusprojekti-opintojaksolla läpivietävän ohjelmistoprojektin kaltaisiin käyttötilanteisiin.

Mallinnusvälineiden kartoitusta, arviointia ja vertailua varten on selvitettävä, mihin ja miten mallinnusvälinettä käytetään Sovellusprojekti-opintojaksolla. Vertailukohteita valittaessa keskitytään mallinnusvälineiden UML-tukeen, käytettävyyteen ja elinkaarikustan-

nuksiin. Vertailukohteita sovelletaan arvioitaessa vertailtaviksi valittujen avoimen lähdekoodin ohjelmistojen sopivuutta Sovellusprojektien käyttötilanteisiin, sekä niiden etuja ja puutteita suhteessa verrokkina käytettyyn suljetun lähdekoodin ohjelmistoon.

Arviointikohteiden ja ohjelmistojen vertailun tulokset voivat olla jossain määrin hyödynnettävissä myös ohjelmistoja kehittävissä mikroyrityksissä. Tarkempaa kartoitusta UML-mallinnusvälineiden käytöstä tai niiden tukemista tehtäväkokonaisuuksista mikroyrityksissä ei kuitenkaan ole tämän tutkielman yhteydessä mahdollista tehdä. Mikroyritysten toimintaympäristöstä tehtyjä oletuksia käsitellään tarkemmin luvussa 2.2.

Ohjelmistokehitystä tukevia ohjelmistoja on kehitetty moniin eri tarkoituksiin, kuten ohjelmistojen suunnitteluun, versiohallintaan, testaukseen, dokumentointiin ja projektihallintaan. Tutkielman aihepiiriä on työmäärän rajaamiseksi tarkennettu siten, että siinä keskitytään tarkastelemaan UML-mallien ja -kaavioiden laatimiseen, käsittelyyn ja varastointiin soveltuvia yleiskäyttöisiä graafisia mallinnusvälineitä. Yleiskäyttöisyys tarkoittaa tutkielmassa sitä, että mallinnusväline ei esimerkiksi pakota käyttäjää seuraamaan mitään tiettyä ohjelmistokehityksen prosessimallia. Graafisuudella tarkoitetaan sitä, että käyttäjän tulee voida piirtää ja muokata kaavioita piirtotyökaluja käyttäen, eikä esimerkiksi kirjoittamalla OCL-kielistä koodia. Lisäksi työmäärää on rajattu valitsemalla tutkielmaan kolme verrattavaa avoimen lähdekoodin ohjelmistoa ja yksi suljetun lähdekoodin ohjelmisto.

3 UML-mallinnuskieli

Ohjelmistokehityksessä kuvataan kohteena olevaa ohjelmistoa ja siihen liittyvää ongelma-aluetta yleisesti mallien avulla. Malleja hyödynnetään eri tavoin ohjelmistokehityksen eri vaiheissa. Esimerkiksi määrittelyvaiheessa mallit toimivat apuna määritettäessä sidosryhmien ohjelmistolle asettamat vaatimukset, ja toteutusvaiheessa ne ohjaavat ohjelmoijien työtä. Mallien esittämiseen käytetään erilaisia kuvaustapoja, kuten kaavioita, tekstiä tai taulukoita. Jos kuvaustapa on täsmällisesti määritelty, voidaan sitä kutsua mallinnuskieleksi.

Unified Modeling Language (UML) on yleiskäyttöinen graafinen mallinnuskieli, jota käytetään olioperustaisessa ohjelmistokehityksessä tietojärjestelmän osien ja rakenteen määrittämiseen, suunnitteluun, visualisointiin ja dokumentointiin. Useissa lähteissä, kuten [Kollmann, s. 1] ja [Sommerville, s. 155], todetaan UML:n saavuttaneen olio-suuntautuneessa ohjelmistotuotannossa käytännön standardin aseman.

Luku perustuu osin tekijän kandidaatin tutkielmaan [Rautiainen]. Luvun sisältöä on siihen verrattuna täydennetty lisäämällä tutkielman vertailukohteisiin liittyviä seikkoja, sekä UML-standardin versiota 2.0 edeltäneiden versioiden käsittelyä, koska osa tarkastelluista mallinnusvälineistä perustuu näihin versioihin. Toisaalta on karsittu muutamia tämän tutkielman kannalta tarpeettomia kohtia. Tekstiä on myös kirjoitettu uudelleen erityisesti kaaviotyyppien osalta, ja sovitettu tähän tutkielmaan sopivaksi.

Koska tutkielmassa arvioitavat mallinnusvälineet käytännössä tukevat eri UML-standardin versioita, on versioiden oleelliset erot syytä tunnistaa. UML:n kehitystä ja eri versioita käsitellään luvussa 3.1. Luvussa 3.2 esitellään UML-mallinnuksen käsitteitä ja luvussa 3.3 UML:n kaaviotyypit. Luvussa 3.4 tarkastellaan UML:n hyödyntämistä ohjelmistokehityksessä.

3.1 UML:n historiaa

UML:a edelsi joukko erilaisia olioperustaisia suunnittelumenetelmiä, joissa esitettyjä ajatuksia ja merkintätapoja yhdistelemällä se on kehitetty. UML:n kehitystä on kuvattu luvussa 3.1.1. Luvussa käsitellään myös UML:n standardointia.

UML:sta on sen ensimmäisen version jälkeen julkaistu useita korjauksia ja uusia ominaisuuksia sisältäneitä versioita. Näiden versioiden kehitystä on käsitelty luvussa 3.1.2.

3.1.1 UML:n kehitys ja standardointi

Perinteisiä **proseduraalisia ohjelmointikieliä**, kuten Cobolia ja Fortrania, tukevat **määrittely- ja suunnittelumenetelmät** kehitettiin 1970-luvulla. Näiden menetelmien käyttö yleistyi 1980-luvun aikana.

Olioperustainen ohjelmointi alkoi Fowlerin [Fowler, s. 7] mukaan levitä tutkimuslaitosten ulkopuolelle 1980-luvulla Smalltalk- ja C++-ohjelmointikielten menestyksen myötä. **Olioperustaisten määrittely- ja suunnittelumenetelmien** lähtökohdan muodostavat teokset julkaistiin vuosien 1988 ja 1992 välisenä aikana. Näistä ensimmäisinä ilmestyivät Sally Shlaerin ja Steve Mellorin sekä Peter Coadin ja Ed Yourdonin kirjat. Pian tämän jälkeen julkaisivat teoksissaan Grady Booch OOAD-menetelmän, Jim Rumbaugh työryhmineen OMT-menetelmän ja Smalltalk-yhteisö oman menetelmänsä. Maininnan arvoinen on myös Ivar Jacobsonin vuonna 1992 julkaisema Objectory-menetelmä. Se perustui osin aiempiin menetelmiin, mutta siinä näkökulma keskittyi niistä poiketen käyttötapauksiin ja ohjelmistokehitysprosessiin.

Fowler toteaa kirjassaan [Fowler, s. 7], että 1990-luvun alussa julkaistuissa lukuisissa oliopohjaisia suunnittelumenetelmiä käsittelevissä kirjoissa kirjoittajat käyttivät toisistaan poikkeavia käsitteistöjä, määrittelyjä, merkitsemistapoja, termistöjä ja prosesseja. Yleisesti kirjoittajat kuitenkin tarkoittivat erilaisilla termeillä tosiaan vastaavia rakenteita, vaikka joissain teoksissa esitettiinkin uusia hyödyllisiä käsitteitä. Jo tuolloin yritettiin olioperustaisten suunnittelumenetelmien yhdistämistä. Esimerkiksi Coleman työryhmineen pyrki yhdistämään Fusion-menetelmään OMT-, Booch- ja CRC-menetelmien käsitteet.

Kirjassa [Booch, s. xvii–xx] kuvataan vuonna 1994 käynnistynyttä ensimmäistä onnistunutta suunnittelumenetelmien yhdistämistä. Tuolloin OMT-menetelmän pääkehittäjä Rumbaugh palkattiin Rational Software Corporation -yhtiöön, jossa Booch työskenteli. He alkoivat yhdistää OMT- ja Booch-menetelmien käsitteistöä Unified Method -menetelmäksi, josta julkaistiin ensimmäinen ehdotus vuonna 1995. Samaan aikaan Objectory- ja OOSE-

menetelmien kehittäjä Ivar Jacobson siirtyi Rational Softwaren palvelukseen sen ostettua ruotsalaisen Objective Systemsin. Nimi Unified Modeling Language otettiin Erikssonin [Eriksson, s. 4] mukaan käyttöön kehittäjien havaittua, että menetelmän sijasta he itse asiassa kehittivät mallinnuskieltä.

Object Management Group (OMG) pyysi vuonna 1996 ehdotuksia oliomallinnusstandardiksi. OMG:n mukaantulon jälkeen Booch, Jacobson ja Rumbaugh alkoivat kehittää sen jäsenistölle soveltuvaa menetelmää ja kuvauskieltä yhdessä muun muassa Hewlett-Packardin, IBM:n, Microsoftin ja Oraclen edustajien kanssa.

Myös useita kilpailevia standardiehdotuksia oli kehitteillä samaan aikaan, mutta ne kaikki yhdistyivät lopulliseen UML-ehdotukseen, joka toimitettiin kirjan [Booch, s. xx] mukaan OMG:lle heinäkuussa 1997. OMG:n jäsenet hyväksyivät UML:n standardiksi marraskuussa 1997, ja samalla OMG otti vastuulleen UML:n jatkokehityksen.

3.1.2 UML:n versiot

Ensimmäinen julkinen UML-versio on lokakuussa 1995 julkaistu Unified Methodin versio 0.8. Vuonna 1996 julkaistiin versiot 0.9 ja 0.91. Tammikuussa 1997 OMG:lle toimitettu ehdotus on UML:n versio 1.0. OMG:n marraskuussa 1997 hyväksymä versio on UML 1.1, josta OMG kuitenkin päätti käyttää versionumeroa 1.0. Kirjan [Fowler, s. 151] mukaan standardoituun versioon viitataan kuitenkin käytännössä numerolla 1.1.

UML:n versio 1.2 julkaistiin vuonna 1998, versio 1.3 vuonna 1999, versio 1.4 vuonna 2001 ja versio 1.5 vuonna 2002. Näiden versioiden muutokset kohdistuivat pääasiassa UML:n sisäisiin rakenteisiin. Versiossa 1.3 tehtiin myös näkyviä muutoksia erityisesti käyttötapaus- ja aktiviteettikaavioihin, sekä versioon 1.4 lisättiin luvussa 3.2.2 esiteltävä profiilimekanismi.

Vuonna 2003 julkaistiin UML:sta tähän mennessä eniten uudistettu versio numerolla 2.0. Näkyvin muutos verrattuna aikaisempiin versioihin ovat uudet kaaviotyypit. UML 2.0:ssa on myös esimerkiksi parannettu rakennetta ja käyttäytymistä kuvaavien kaavioiden yhteyttä toisiinsa sekä olennaisesti laajennettu sekvenssikaavion ilmaisuvoimaa. Version 2.0 jäl-

keen on julkaistu useita versioita, jotka sisältävät pieniksi luonnehdittuja korjauksia versioon 2.0. Viimeisin niistä on vuonna 2010 julkaistu versio 2.4.1.

Tutkielmassa käytetään UML:n versiosta 1.1–1.5 nimitystä UML 1 ja versioista 2.0–2.4.1 nimitystä UML 2, kun viitataan yleisesti näihin versioihin. Taulukossa 1 on kuvattu luvussa esitellyt kaaviotyypit pääversioittain. UML-versiossa 1 käytettävissä oli 10 kaaviotyyppiä, ja UML-versiossa 2 kaaviotyyppien määrää lisättiin neljällä. Yhteistyökaavio on nimetty UML-versiossa 2.0 yhteistoimintakaavioksi.

Kaaviotyyppi	**Kuvaus**	**Versiossa**
Komponenttikaavio	Kuvaa komponenttien rakennetta ja yhteyksiä.	UML 1
Käyttötapaus	Kuvaa, kuinka käyttäjä on vuorovaikutuksessa järjestelmän kanssa.	UML 1
Luokkakaavio	Kuvaa luokkia, niiden ominaisuuksia ja suhteita.	UML 1
Sekvenssikaavio	Kuvaa olioiden välistä vuorovaikutusta painottaen toimintojen etenemistä.	UML 1
Sijoittelukaavio	Kuvaa artefaktien sijoittumista solmuihin.	UML 1
Tilakaavio	Kuvaa, kuinka tapahtumat muuttavat oliota sen elinaikana.	UML 1
Aktiviteettikaavio	Kuvaa prosesseja ja rinnakkaista toimintaa.	UML 1
Oliokaavio	Kuvaa (esimerkein) ilmentymien ohjelman suorituksenaikaista kokoonpanoa.	UML 1 (epävirallisesti)
Pakkauskaavio	Kuvaa käännöksen aikaista hierarkkista rakennetta.	UML 1 (epävirallisesti)
Ajoituskaavio	Kuvaa olioiden välistä vuorovaikutusta painottaen ajoitusta.	UML 2
Kokoava vuorovaikutuskaavio	On aktiviteetti- ja sekvenssikaavion yhdistelmä.	UML 2
Koostekaavio	Kuvaa luokan ajonaikaista jakautumista.	UML 2
Profiilikaavio	Kuvaa profiileja, stereotyyppejä ja niihin liittyviä metaluokkia.	UML 2
Yhteistoimintakaavio (Yhteistyökaavio)	Kuvaa olioiden välistä vuorovaikutusta painottaen yhteyksiä.	UML 2 (UML 1)

Taulukko 1. Kaaviotyypit UML-versioittain kirjan [Fowler, s. 11] mukaan.

3.2 UML-mallinnus

Suunnittelun kohteena olevaa ohjelmistoa ja siihen liittyvää kohdealuetta kuvataan ohjelmistokehityksessä yleisesti erilaisten mallien avulla. Mallissa kohteen tarkastelua pyritään yksinkertaistamaan kuvaamalla järjestelmästä vain valitun asiakokonaisuuden ja näkökulman käsittelyn kannalta oleelliset osat ja poistamalla turhia yksityiskohtia.

Malleja voidaan käyttää paitsi suunniteltavan järjestelmän vaatimusten, ymmärtämisen ja rajaamisen apuna, myös määrittelyjen ja dokumenttien laatimiseen, sekä olemassa olevien järjestelmien toteutuksen kuvaamiseen. Ohjelmistoa kehitettäessä ne toimivat ihmisten välisen kommunikaation tukena.

3.2.1 Mallinnuksen peruskäsitteitä

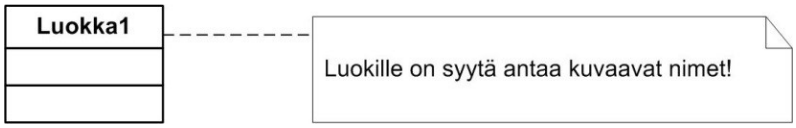
Malli (engl. *model*) on kohteena olevasta tietojärjestelmästä laadittu kuvaus, joka kuvaa järjestelmästä valitun asiakokonaisuuden käsittelyn kannalta oleelliset osat. Malleja käytetään kohteen ymmärtämisen ja rajaamisen apuna, sekä määrittelyjen ja dokumenttien laatimiseen. Mallien esittämiseen käytetään erilaisia kuvaustapoja, kuten graafisia kaavioita, tekstiä tai taulukoita. **Mallinnuskielestä** (engl. *modeling language*) voidaan puhua, kun kuvaustapa on täsmällisesti määritelty. Määrittelyn tulisi kuvata kielen rakenne ja rakenteiden merkitys.

UML on graafinen mallinnuskieli, jonka osakielet on koottu useista sitä edeltäneiden mallinnuskielten kuvaustavoista. UML:en sisältyviä osakieliä kutsutaan **kaaviotyypeiksi**.

UML-standardi määrittelee kielen rakenteen notaation ja metamallin (engl. *metamodel*) avulla. **Notaatio** on mallinnuskielen syntaksi, joka kuvaa malleissa käytettävän graafisen esitystavan. **Metamalli** kuvaa mallien sallitut abstraktit rakenteet ja määrittelee joukon nk. **hyvinmuodostuneisuussääntöjä** (engl. *well-formedness rules*), jotka edelleen rajoittavat sallittuja UML-malleja.

Standardin [OMG 2011b, s. 691] mukaan UML:n mallit koostuvat **mallinnuselementeistä** (engl. *model element*), jotka vastaavat yleisiä olio-ohjelmoinnin käsitteitä, kuten luokka (engl. *class*), sekä niiden välisiä suhteita, kuten assosiaatio (engl. *association*). Jokaisella mallinnuselementillä on sitä vastaava graafinen symboli, eli **näkymäelementti** (engl. *view*

element). Ne ovat geometrisiä kuvioita (kuten suorakaiteita kuvan 1 tapaan), ja niiden väliset suhteet kuvataan erityyppisinä suhdeviivoina. Samaa mallinnuselementtiä voidaan käyttää useissa kaaviotyypeissä, mutta sen merkitys ja ulkoasu pysyvät muuttumattomina.



Kuva 1: Luokkaan on liitetty huomautus.

Kaaviot (engl. *diagram*) ovat mallin osien graafisia esityksiä, jotka koostuvat näkymäelementeistä. Niillä kuvataan järjestelmän korkean tason toiminnallisuutta, staattista ja dynaamista rakennetta sekä toiminnan aikaista käyttäytymistä. Kukin kaavio noudattaa kaaviotyyppinsä määrittelemää rakennetta.

Standardissa [OMG 2011a, s. 4] todetaan, että näkymäelementtien ja suhdeviivojen ulkoasun määrittely jää osin työkaluohjelmistojen valmistajien päätettäväksi. Ohjelmisto voi myös jättää näyttämättä mallista tiettyjä asioita käyttäjän valinnan mukaan tai automaattisesti. Yhdestä mallista voi näin olla useita eri käyttötarkoituksiin sopivia erilaisia näkymiä.

Mallinnuselementtien **ominaisuuksien** (engl. *properties*) avulla esitetään elementin tietoja. Ominaisuus voi olla esimerkiksi luokan tarkoituksen ja toiminnot kuvaava lyhyt vapaamuotoinen teksti.

Kaavioihin voidaan liittää mallinnuselementtien perusominaisuuksiin kuulumatonta lisäinformaatiota yhteisillä merkinnöillä, joita ovat koristeet, huomautukset ja lisätietomääreet. **Lisätietomääreitä** (engl. *tagged value*) käsitellään luvussa 3.2.2.

Mallinnuselementteihin voidaan liittää **koristeita** (engl. *adornments*), jotka kirjan [Booch, s. 27–28] mukaan lisäävät näkymäelementtiin uutta informaatiota. Niitä voidaan käyttää esimerkiksi luokkaa kuvaavassa näkymäelementissä ilmaisemaan attribuuttien ja metodien näkyvyys.

Huomautuksen (engl. *note*) avulla voidaan kuvan 1 tapaan lisätä malliin tietoa, jota siinä ei voida muuten esittää. Se voidaan sijoittaa minne tahansa missä tahansa kaaviossa, ja se voi sisältää mitä tahansa tietoa, kuten kommentin tai viittauksen erilliseen dokumenttiin.

3.2.2 UML:n laajentaminen ja sovittaminen

Standardissa [OMG 2011b, s. 660] esitetään useita syitä, joiden vuoksi UML:n metamallia voi olla tarpeen laajentaa. Esimerkiksi voidaan haluta lisätä tiettyyn menetelmään liittyvää termistöä tai muuttaa jonkin symbolin esitystapaa tietylle organisaatiolle paremmin sopivaksi.

Profiilit (engl. *profile*) ovat standardin [OMG 2011b, s. 659] mukaan mekanismi, jolla UML-metamallin laajentaminen on mahdollista. Niiden avulla sovittaminen tiettyyn tarkoitukseen voidaan tehdä muuttamatta itse mallinnuskieltä. Profiilit lisättiin Fowlerin kirjan [Fowler, s. 157] mukaan UML-standardin versioon 1.4. Profiiliin voidaan paketoida yhteenkuuluvat, esimerkiksi tiettyyn liiketoiminta-alueeseen liittyvät, UML:n metamallin laajennukset, jolloin suunnittelija voi ottaa ne käyttöönsä yhtenä kokonaisuutena.

Jo ennen profiilien lisäämistä oli UML:n laajentaminen mahdollista standardin [OMG 2011b, s. 659] mukaan UML:n versiosta 1.1 alkaen käyttämällä lisätietomääreitä ja stereotyyppejä. Myöhemmissä UML-versioissa stereotyyppien ja lisätietomääreiden määritelmiä on täsmennetty profiili-käsitteen avulla.

Stereotyyppi (engl. *stereotype*) kuvaa, kuinka mallinnuselementtiä voidaan laajentaa. Stereotyyppi laajentaa perustana olevan mallinnuselementin merkitystä, muttei muuta sen rakennetta. Kaaviossa stereotyyppi voidaan esittää omalla symbolillaan tai sijoittamalla näkymäelementissä sen nimi kaksoiskulmasuluissa «» elementin nimen eteen tai yläpuolelle. UML-standardi [OMG 2011b, s. 703–709] määrittelee joukon valmiita käytössä vakiintuneita stereotyyppejä, joiden lisäksi on mahdollista määritellä omia stereotyyppejä.

Mallinnuselementteihin voidaan liittää **lisätietomääreitä** (nimetty arvo, merkkaus, engl. *tagged value*), joiden avulla niihin voidaan kiinnittää esimerkiksi koodigeneraattorin tai prosessiohjauksen tarvitsemia tietoja. UML:n versiossa 1.3 oli mahdollista liittää lisätietomääreitä suoraan mallinnuselementteihin. Standardin [OMG 2011b, s. 686] mukaan

UML:n versiossa 2 lisätietomäärettä voidaan käyttää vain, jos se on määritelty stereotyypissä. Toisin sanoen mallinnuselementtiä täytyy laajentaa stereotyypillä ennen kuin siihen voidaan liittää lisätietomääre. Mallinnustyökalu voi tosin piilottaa stereotyypin lisäämisen käyttäjältä, jolloin lisätietomääreen liittäminen muistuttaa UML:n version 1.3 mukaista toimenpidettä.

UML:n sovittamiseen voidaan käyttää myös **rajoitteita** (engl. *constraint*). Ne rajaavat joko mallinnuselementin käyttöä tai sen merkitystä. Niiden avulla mallin laatijaa voidaan ohjata käyttämään mallinnuselementtiä tarkoitetulla tavalla. Kaavioissa rajoitukset näytetään aaltosulkujen sisällä. UML-standardissa on määritelty joukko valmiita rajoitteita, joiden lisäksi on mahdollista määrittää omia rajoitteita.

3.3 Kaaviotyypit

Tarkasteltavaa tietojärjestelmää kuvaava malli esitetään UML:ssa kaavioiden avulla. Kaaviot ovat piirroksia, joissa mallinnuselementit on järjestetty kuvaamaan järjestelmää tietystä näkökulmasta. Järjestelmän kokonaiskuva muodostetaan yhdistämällä eri kaaviotyyppien mukaisia kaavioita.

UML:n versiossa 1 on 10 ja versiossa 2 on 13 kaaviotyyppiä. UML-standardin versioon 2.2 on lisätty 14. kaaviotyypiksi profiilikaavio. Kaaviotyypit on lueteltu UML-versioittain taulukossa 1 luvussa 3.1.2.

Kirjassa [Sommerville, s. 329] todetaan, että tietojärjestelmää mallinnettaessa on yleensä huomioitava kaksi toisiaan täydentävää ja toisistaan riippuvaa näkökulmaa, jotka ovat järjestelmän rakenne ja sen toiminnan aikainen käyttäytyminen. UML:n kaaviotyypit on standardissa [OMG 2011b, s. 694] jaettu näiden näkökulmien perusteella rakennekaavioihin ja käyttäytymiskaavioihin.

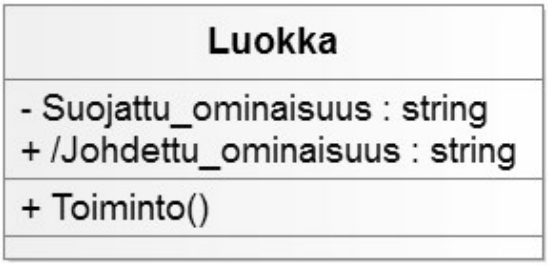
Rakennekaaviot kuvaavat järjestelmän staattista, ajasta riippumatonta, rakennetta ja **käyttäytymiskaaviot** järjestelmän ajonaikaista, ajasta riippuvaa, rakennetta. Käyttäytymiskaavioista voidaan edelleen erottaa vuorovaikutusta kuvaavat kaaviotyypit omaksi ryhmäkseen. Rakennekaaviot on kuvattu luvuissa 3.3.1–3.3.3, käyttäytymiskaaviot luvuissa 3.3.4–3.3.6 ja vuorovaikutuskaaviot luvuissa 3.3.7–3.3.8.

Standardissa huomautetaan, että kaaviotyyppien rajat eivät ole joustamattomia, vaan usein on sallittua käyttää kaaviossa toisen kaaviotyypin mallinnuselementtejä. Myös rakenne- ja käyttäytymiskaavioiden merkintöjä voidaan tarvittaessa käyttää samassa kaaviossa.

3.3.1 Luokkakaavio

Luokkakaavio (engl. *class diagram*) on rakennekaavio, joka kuvaa järjestelmän staattista rakennetta luokkien ja niiden välisten yhteyksien avulla. Kaaviossa voidaan esittää tiettyjä toiminnallisia elementtejä (kuten luokan toimintoja), mutta niiden dynaaminen toiminta kuvataan muiden kaavioiden avulla. Luokkakaaviota voidaan hyödyntää ohjelmiston luokkien esittämisen lisäksi myös mallinnettavan sovellusalueen käsitteistön kuvaamisessa. Kirjassa [Booch, s. 105] todetaan luokkakaavion olevan yleisin oliomallinnuksessa käytetyistä rakennekaaviosta. Se on myös vuonna 2008 julkaistun kyselytutkimuksen [Dobing, s. 5] perusteella eniten käytetty UML-kaaviotyyppi.

Standardin [OMG 2011b, s. 49–50] mukaan luokka esitetään kaaviossa kuvan 2 tapaan suorakaiteena, jonka sisällä ovat omissa osioissaan luokan nimi, luokan ominaisuudet ja toimintojen otsaketiedot. Ominaisuus- ja toiminto-osio voidaan piilottaa ja niiden lisäksi voidaan joissain välineissä näyttää näkymäelementissä lisäosio esimerkiksi kommentteja varten. Standardissa annetaan muutamia **luokan näkymäelementin** ulkoasuun liittyviä ohjeita, esimerkiksi kehotetaan kirjoittamaan abstraktin luokan nimi kursiivilla. Aktiivinen luokka ilmaistaan näkymäelementissä suorakaiteen sivureunojen kahdennetulla viivalla.

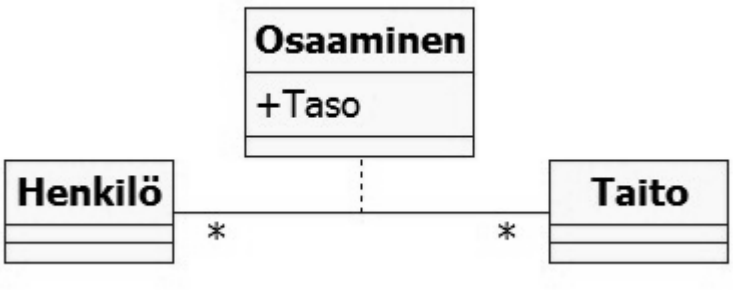


Kuva 2: `Luokka`-näkymäelementti Modelio-ohjelmistolla piirrettynä.

Luokan ominaisuuksista ja toiminnoista käytetään UML-terminologiassa yleisnimitystä **piirre** (engl. *feature*). Luokan näkymäelementissä voidaan ilmaista ominaisuuksien näkyvyys merkkejä tai graafisia symboleita käyttäen. Muiden ominaisuuksien perusteella **joh-**

dettu ominaisuus voidaan ilmaista lisäämällä kauttaviiva ominaisuuden nimen eteen kuvassa 2 esitetyllä tavalla. Yksittäisen luokan ilmentymän sijasta luokkaan liittyvät staattiset toiminnot ja ominaisuudet ilmaistaan alleviivaamalla piirteen nimi.

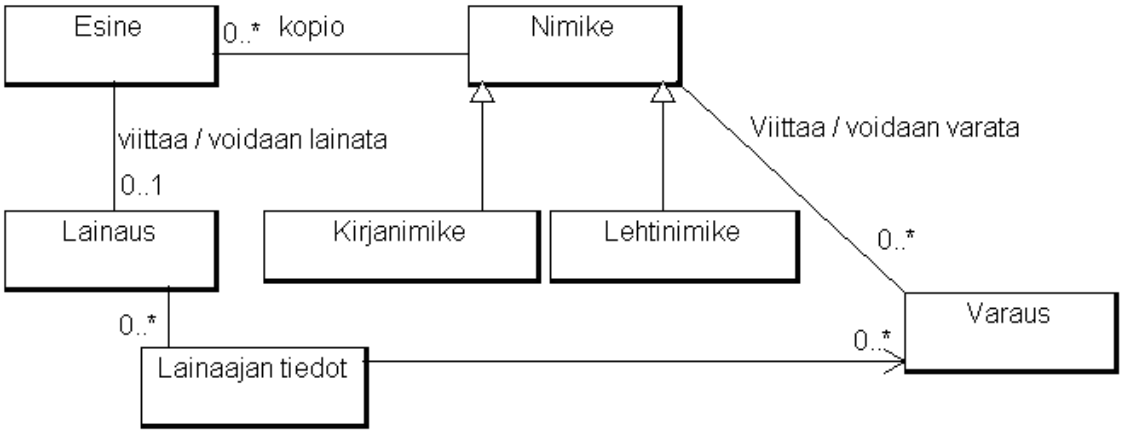
Luokkien väliset yhteydet esitetään suhteita kuvaavien näkymäelementtien avulla. **Assosiaatio** kuvataan suoralla viivalla, jonka suunta voidaan tarvittaessa osoittaa avopäisellä nuolella. Luokkien väliseen assosiaatiosuhteeseen voidaan liittää **assosiaatioluokka** (engl. *association class*). Esimerkiksi kuvassa 3 assosiaatioluokan avulla ilmaistaan, että kuhunkin `Henkilö`-luokan `Taito`-luokkaan voi liittyä yksi `Osaaminen`-luokan `Taso`.



Kuva 3: Assosiaatioluokka WhiteStarUML-ohjelmistolla piirrettynä.

Yleistyssuhde (engl. *generalization*) kuvataan käyttäen nuolta, jonka kärkenä on avoin kolmio. **Riippuvuussuhdetta** (engl. *depency*) kuvaavan nuolen kärkenä on täytetty kolmio. **Koostetta** (engl. *aggregation*) kuvaavan viivan päässä on avoin vinoneliö ja **vahvaa koostetta** (engl. *composition*) kuvaavan viivan päässä täytetty vinoneliö.

Kaaviossa voidaan kuvata sekä assosiaatioiden että ominaisuuksien **kerrannaisuus** (engl. *multiplicity*). Kerrannaisuuden merkitsemiseen käytetään rajoittamatonta arvoa merkitsevää `*`-merkkiä sekä kokonaislukuja. Kerrannaisuus määritellään ilmaisemalla kerrannaisuuden mahdolliset ylä- ja alarajat. Esimerkiksi kuvassa 4 oleva merkintä `0..1` tarkoittaa, että `Esine`-luokan olio voidaan liittää `Lainaus`-luokan olioon korkeintaan kerran ja `0..*` -merkintä `Nimike` ja `Varaus`-luokkien välisessä assosiaatiossa tarkoittaa, että nimikkeellä voi olla rajoittamaton määrä varauksia tai ei yhtään varausta. Kuvassa 3 käytetty merkintä `*` on standardissa [OMG 2011b, s. 129] kuvattu lyhennetty muoto rajoista `0..*`.



Kuva 4: Esimerkkijärjestelmään liittyvä luokkakaavio ArgoUML-ohjelmistolla piirrettynä.

Joissain olio-ohjelmointikielissä on käytössä **mallinneluokka**-käsite (engl. *parameterized class* tai *template*). Mallinneluokka esitetään kaaviossa luokkanäkymäelementillä, jonka oikeassa yläkulmassa on katkoviivalla kehystetty T-kirjain.

Enumeraatio on kiinteä arvojoukko. Se esitetään luokkana, jonka nimeen on liitetty `enumeration`-avainsana ja jonka ominaisuusosiossa luetellaan siihen kuuluvat arvot.

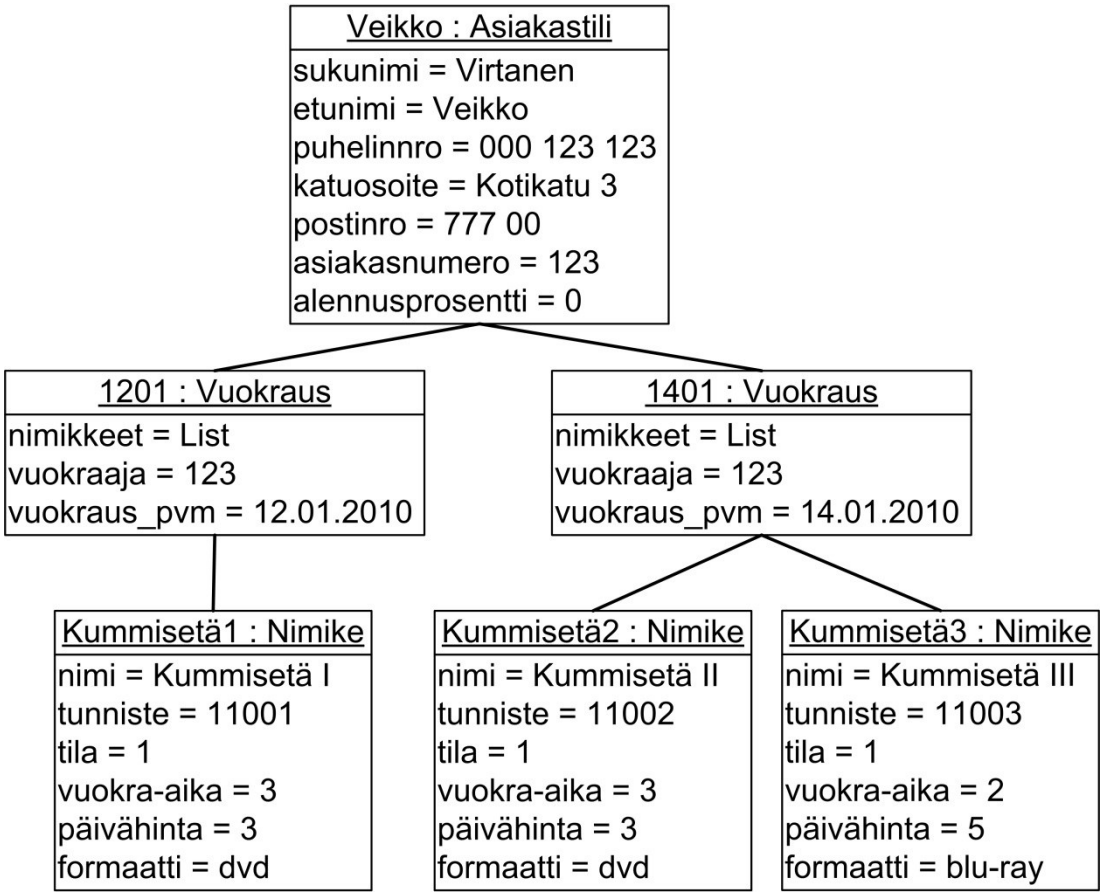
Luokka voi tarjota muille luokille toimintoja rajapinnan kautta. **Luokan tarjoama rajapinta** ilmaistaan avoimena ympyränä, joka on yhdistetty viivalla luokkaan. Luokan tarvitsema rajapinta taas ilmaistaan puolikaarena, joka yhdistetään viivalla luokkaan.

3.3.2 Oliokaavio

Oliokaavio (engl. *object diagram*) on rakennekaavio, johon sisältyy olioiden kuvauksia. Siinä käytetään standardin [OMG 2011b, s. 84] mukaan samoja merkintätapoja ja suhteita kuin luokkakaaviossa. Olio esitetään kuvan 5 tapaan luokkamallinnuselementillä, jossa olion nimi on alleviivattu.

Kirjassa [Booch, s. 195] oliokaavion todetaan olevan kuva järjestelmästä tietyllä ajan hetkellä. Tarkan määrittelyn sijasta oliokaavion järjestelmästä antama kuva toimii esimerkkinä sen toiminnasta, jolloin oliokaavion avulla voidaan muun muassa havainnollistaa mo-

nimutkaisia tietorakenteita. Osa olioiden tiedoista voidaan kuvata epätäydellisesti esimerkiksi antamalla raja-arvot, joiden välillä ominaisuuden arvo voi vaihdella.



Kuva 5: Oliokaavio.

3.3.3 Komponentti-, pakkaus-, kooste-, sijoittelu ja profiilikaavio

Komponentti on itsenäinen ohjelmayksikkö, joka toteuttaa tarkasti määritellyn rajapinnan. **Komponenttikaavio** (engl. *component diagram*) on rakennekaavio, joka esittää standardin [OMG 2011b, s. 145] mukaan järjestelmän komponentit ja niiden väliset yhteydet. Sitä voidaan käyttää havainnollistamaan joko järjestelmän jakoa komponentteihin ja niiden vuorovaikutusta rajapintojen kautta tai haluttaessa paloitella komponentteja alemman tason rakenteiksi. Kirjan [Fowler, s. 141] mukaan komponenttikaavion ja yleisen luokkakaavion raja ei ole tarkasti määritelty, mutta luokkakaaviosta poiketen komponenttikaaviossa kuvataan tyypillisesti arkkitehtuuritason rakenteita. Esimerkiksi kuvassa 6 on esimerkkijärjes-

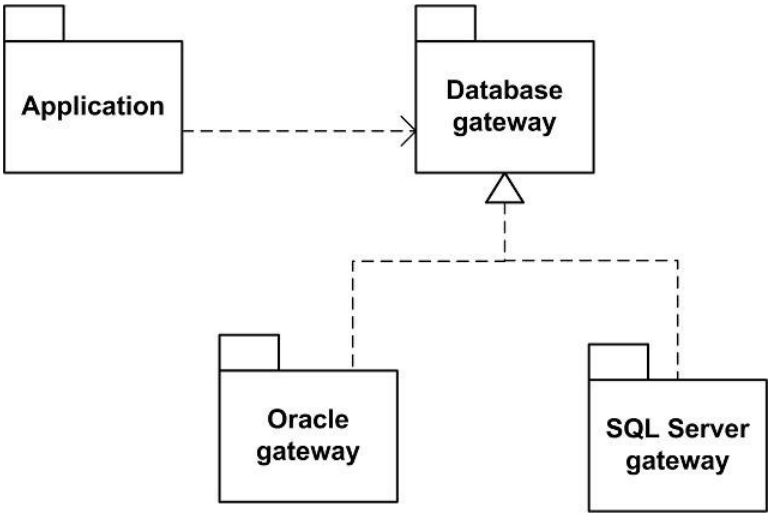
telmän komponenttikaavio, jossa komponenttien näkymäelementit ovat UML 1:n mukaisia.

Standardissa [OMG 2011b, s. 158] esitellään joukko UML:n versiossa 2 komponenttikaavioon tehtyjä muutoksia. Esimerkiksi komponentin näkymäelementin ulkoasu on muuttunut kuvassa 7 esitetyn kaltaiseksi.

Kuva 6: WhiteStarUML-ohjelmistolla piirretty esimerkkijärjestelmän komponenttikaavio.

Kuva 7: UML 2 -version mukaisia `komponentti`-näkymäelementtejä. Vasemmalla oleva on piirretty Rhapsody Modeler ja oikeanpuoleinen Modelio-ohjelmistolla.

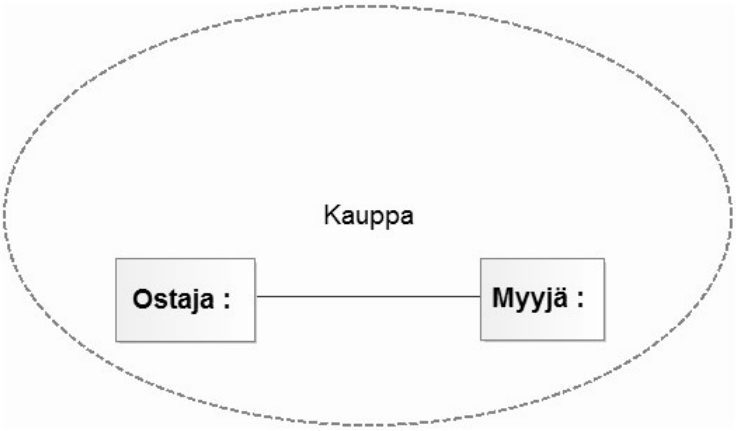
Pakkaus (engl. *package*) on rakenne, jonka avulla mitä tahansa UML:n elementtejä voidaan ryhmitellä korkeamman tason yksiköiksi. **Pakkauskaavio** (engl. *package diagram*) on kuvassa 8 esitetyn kaltainen rakennekaavio, joka koostuu pääasiassa pakkauksista ja niiden välisistä yhteyksistä. Kirjan [Fowler, s. 89] mukaan pakkauskaaviota käytetään yleisimmin luokkien ryhmittelyyn, mutta sen avulla voidaan ryhmitellä myös muita mallinnuselementtejä suuremmiksi kokonaisuuksiksi.



Kuva 8: Pakkauskaavio.

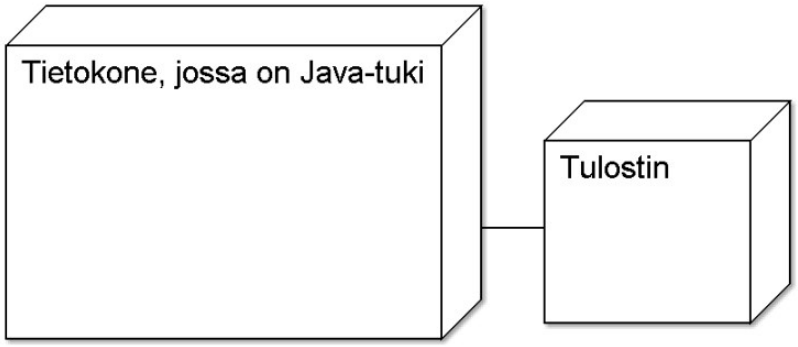
Koostekaavio (engl. *composite structure diagram*) on standardin [OMG 2011b, s. 197] mukaan rakennekaavio, jonka avulla havainnollistetaan yleensä luokan sisäistä rakennetta ja rakenneosien suorituksen aikaista vuorovaikutusta. Oleellinen ero pakkauskaavioon on siis se, että pakkauskaavio kuvaa käännöksen aikaista rakennetta koostekaavion kuvatessa ajonaikaista ryhmittelyä.

Kuvassa 9 on koostekaavion merkinnöin esitetty standardin [OMG 2011b, s. 178] esimerkkiä mukaillen, että `Kauppa` on yhteistoimintaa roolien `Ostaja` ja `Myyjä` välillä. Mallissa `Kauppa`an voitaisiin liittää tarkempi kuvaus vuorovaikutuksen vaiheista esimerkiksi sopivaa vuorovaikutuskaaviota käyttäen.



Kuva 9: Koostekaavio Modelio-ohjelmistolla piirrettynä.

Sijoittelukaavio (engl. *deployment diagram*) tunnetaan myös nimellä **käyttöönottokaavio**. Standardin [OMG 2011b, s. 199] mukaan sijoittelukaavioissa voidaan kuvata järjestelmän laitearkkitehtuuria ja ohjelmiston osien suorituksen jakautumista laitteiden välillä. Kuvassa 10 on esimerkkijärjestelmän yksinkertainen sijoittelukaavio. Järjestelmä voidaan sijoittaa mihin tahansa ympäristöön, jossa on Javaa tukeva tietokone ja siihen liitetty tulostin. Kaavioon olisi voitu kuvata myös järjestelmän osien jakautuminen eri laitteille, mutta se ei ole esimerkkijärjestelmän tapauksessa tarpeen, koska ne sijoittuvat samaan laitteeseen.



Kuva 10: Esimerkkijärjestelmän sijoittelukaavio ArgoUML-ohjelmistolla piirrettynä.

Profiilikaavio (engl. *profile diagram*) on määritelty UML-standardin versiossa 2.0 rakennekaavioksi, jossa kuvataan luvussa 3.2.2 esiteltyjä profiileja, stereotyyppejä ja niihin liittyviä metaluokkia.

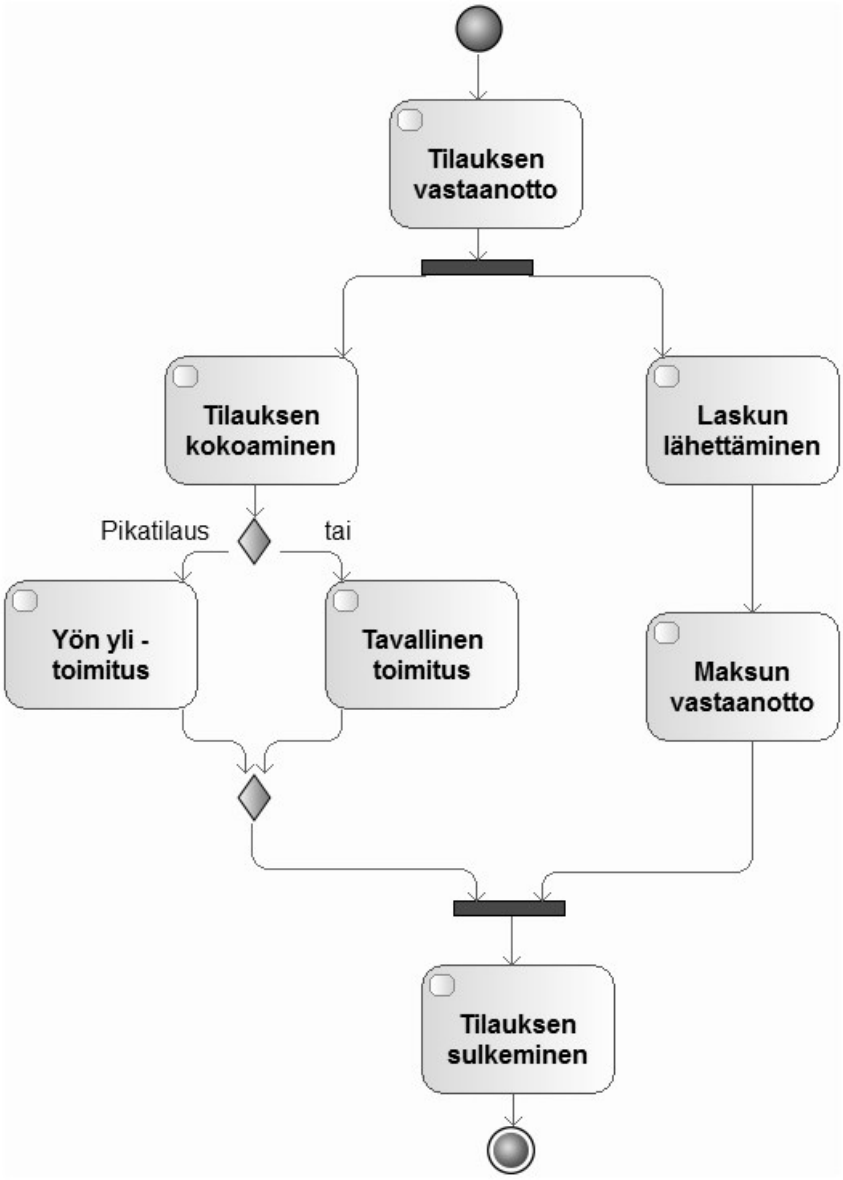
3.3.4 Aktiviteettikaavio

Standardin [OMG 2011b, s. 324] mukaan **aktiviteettikaavio** (engl. *activity diagram*) eli **toimintokaavio** on käyttäytymiskaavio, jolla voidaan kuvata proseduurin toimintaa, liiketoimintaprosessia tai työn kulkua. Kaavio kuvaa **aktiviteetin** (engl. *activity*), joka koostuu **toimenpiteistä** (engl. *action*) ja niiden välisistä siirtymistä. Kaaviossa voidaan esittää myös rinnakkaisia tai vaihtoehtoisesti suoritettavia toimenpiteitä ja olioita.

Useissa UML-versioissa aktiviteettikaavioon on tehty merkittäviä muutoksia. Fowler [Fowler, s. 117] nostaa UML:n versioon 2 tehdyistä muutoksista esille aktiviteettikaavion ja tilakaavion välisen sidoksen poistamisen. Aiemmissa UML:n versiossa aktiviteettikaa-

vion katsottiin olevan tilakaavion erikoistapaus. Yhteys tilakaavioon on koettu ongelmalliseksi etenkin käytettäessä aktiviteettikaaviota työn kulun kuvaamiseen.

Toimenpide esitetään kaaviossa kulmistaan pyöristettynä suorakaiteena kuvan 11 tapaan. Aktiviteetin käynnistävä **alkutila** (engl. *initial node*) kuvataan täytettynä ympyränä ja päättymistä merkitsevä **lopputila** (engl. *activity final*) ympyröidyllä täytetyllä ympyrällä.



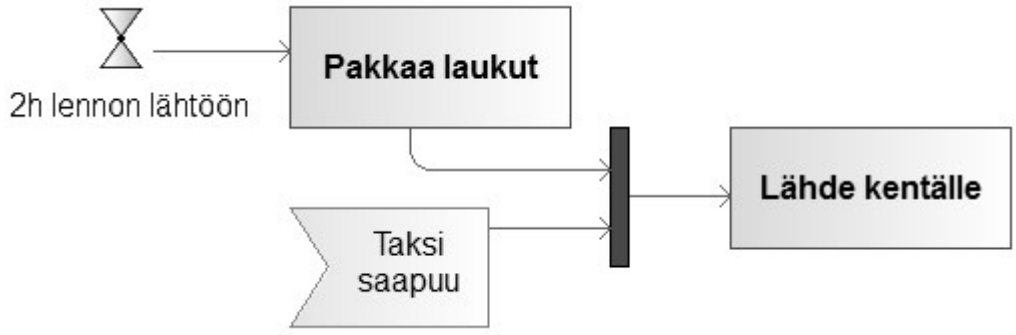
Kuva 11: Aktiviteettikaavio Modelio-ohjelmistolla piirrettynä.

Toimenpiteiden välisiä siirtymiä kuvataan avokärkisillä nuolilla. Siirtymä voi jakautua useiksi eri toimenpiteisiin johtaviksi rinnakkaisiksi siirtymiksi **haarautumispalkkia** (engl. *fork*) käyttäen kuten kuvassa 11. Vastaavasti **liittymispalkkia** (engl. *join*) käyttäen voidaan yhdistää useita siirtymiä. Jos siirtymä ohjautuu päätöksen perusteella eri toimenpiteisiin, käytetään **päätöselementtiä** (engl. *decision*), ja vastaavasti käytetään **yhdistymiselementtiä** (engl. *merge*), kun kaksi siirtymää liittyy yhteen.

Tarvittaessa toimenpiteen sisäinen toiminta voidaan esittää omana erillisenä aktiviteettikaavionaan. Kaavion sisältävään toimenpiteeseen lisätään tällöin **harava**-symboli (engl. *rake*).

Aktiviteettikaaviossa voidaan osoittaa aktiviteettien suorittajat jakamalla kaavio **vyöhykkeisiin** (engl. *partition* tai *swim lane*). UML 1 -versioissa vyöhykkeet jakoivat kaavion pysty- tai vaakasuunnassa. UML 2 -versioissa voidaan käyttää myös standardissa [OMG 2011b, s. 352] kuvattua kaksiulotteista ruudukkojakoa.

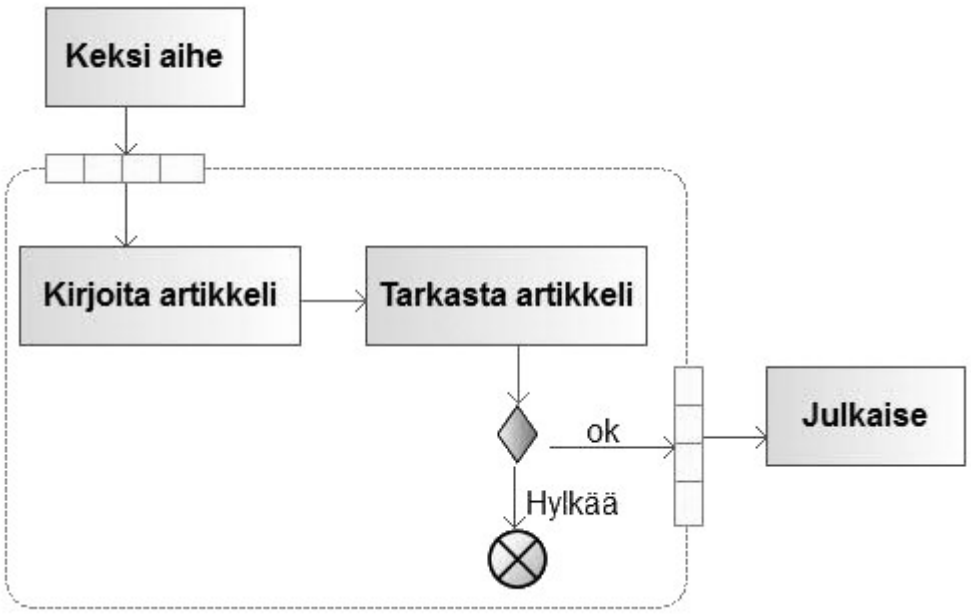
Toimenpiteiden suorittaminen voi riippua myös **signaalien** saapumisesta. Myös signaalien lähettäminen on kaaviossa mahdollista. Kuvassa 12 on esitetty tilanne, jossa toimenpide odottaa kahta signaalia. Laukkujen pakkaaminen alkaa kaksi tuntia ennen lentoa. Ennen kuin voidaan lähteä lentokentälle, on sekä saatava laukut pakattua että taksin saavuttava paikalle.



Kuva 12: Signaaleja kirjasta [Fowler, s. 123] mukaillussa aktiviteettikaaviossa piirrettynä Modelio-ohjelmistolla.

Toimenpiteillä voi olla parametreja samaan tapaan kuin luokkien metodeilla. Parametreja ei ole välttämätöntä kuvata aktiviteettikaaviossa, mutta tarvittaessa niiden esittämiseen voidaan käyttää **tappeja** (engl. *pin*).

Lisäaluetta (engl. *expansion region*) voidaan käyttää tilanteessa, jossa yhden toimenpiteen seurauksena käynnistetään toinen toimenpide useita kertoja. Kuvassa 13 on esitetty, kuinka lisäaluetta käyttäen voidaan kuvata tilanne, jossa useista eri aiheista kustakin kirjoitetaan artikkeli julkaistavaksi. Osa lisäalueen rinnakkaisista tietoimenpiteistä saattaa päättyä, vaikka aktiviteetin suoritus jatkuu. Näissä tilanteissa voidaan päättyminen ilmaista käyttämällä kuvassa 13 näkyvää **tietovuon loppua** (engl. *flow final*) ilmaisevaa symbolia, jossa ympyrän sisään on piirretty vinoristi.



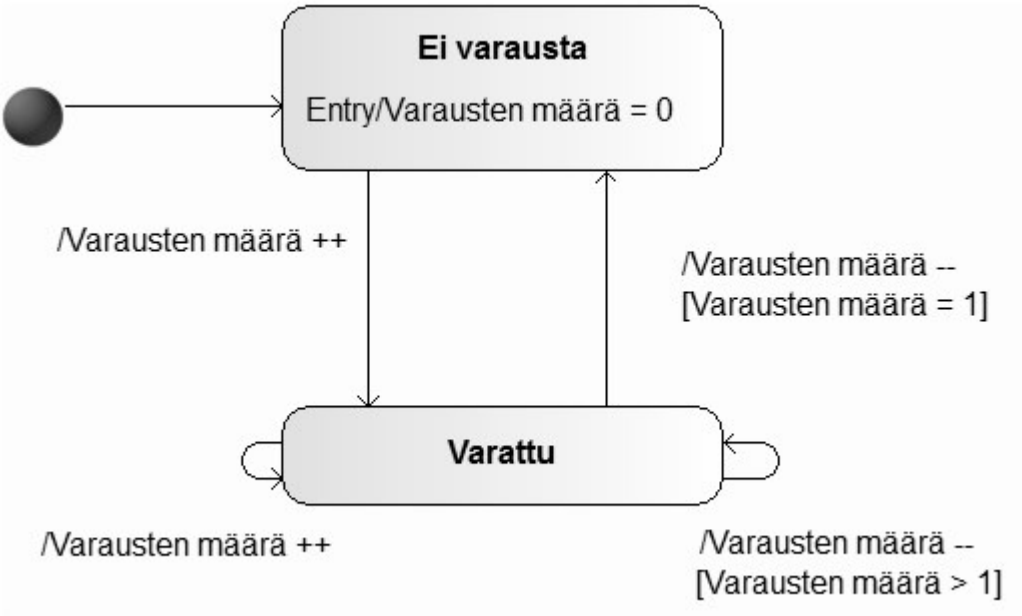
Kuva 13: Lisäalue piirrettynä Modelio-ohjelmistolla kirjan [Fowler, s. 128] kuvan pohjalta.

3.3.5 Tilakaavio

Tilakaavio (engl. *state diagram* tai *state machine diagram*) on standardin [OMG 2011b, s. 535] mukaan käyttäytymiskaavio, jolla voi kuvata tilakoneen avulla ohjelmayksikön (kuten olion) toiminnan tai palveluprotokollan, eli palvelujen kutsujärjestyksen. Kaavio koostuu **tiloista** (engl. *state*), niiden välisistä siirtymistä ja koostetiloista. Tilakaaviossa voi olla yksi alkutila ja useita lopputiloja. Oliosuuntautuneessa lähestymistavassa tilakaaviolla ku-

vataan tyypillisesti luokan yhden olion toiminta. Kaavio esittää kaikki oliolle mahdolliset tilat ja siirtymisen tilojen välillä aiheuttavat **tapahtumat** (engl. *event*).

Tila kuvataan kaaviossa kuvassa 14 esitettyyn tapaan kulmistaan pyöristettynä suorakaiteena. **Alkutilan** näkymäelementti on kuvan mukaisesti täytetty ympyrä ja **lopputilan** ympyröity täytetty ympyrä. Tilan näkymäelementissä voi olla kolme osaa. Nimiosassa on tilan nimi, valinnaisessa muuttujaosassa luetellaan tilan muuttujat ja valinnaisessa toiminto-osassa luetellaan tilan tapahtumat ja toiminnot. Kuvassa 14 on tilan `Ei varausta` toiminto-osassa liitetty tapahtumaan `Entry` toiminto `Varausten määrä = 0`.



Kuva 14: Tilakaavio Modelio-ohjelmistolla piirrettynä.

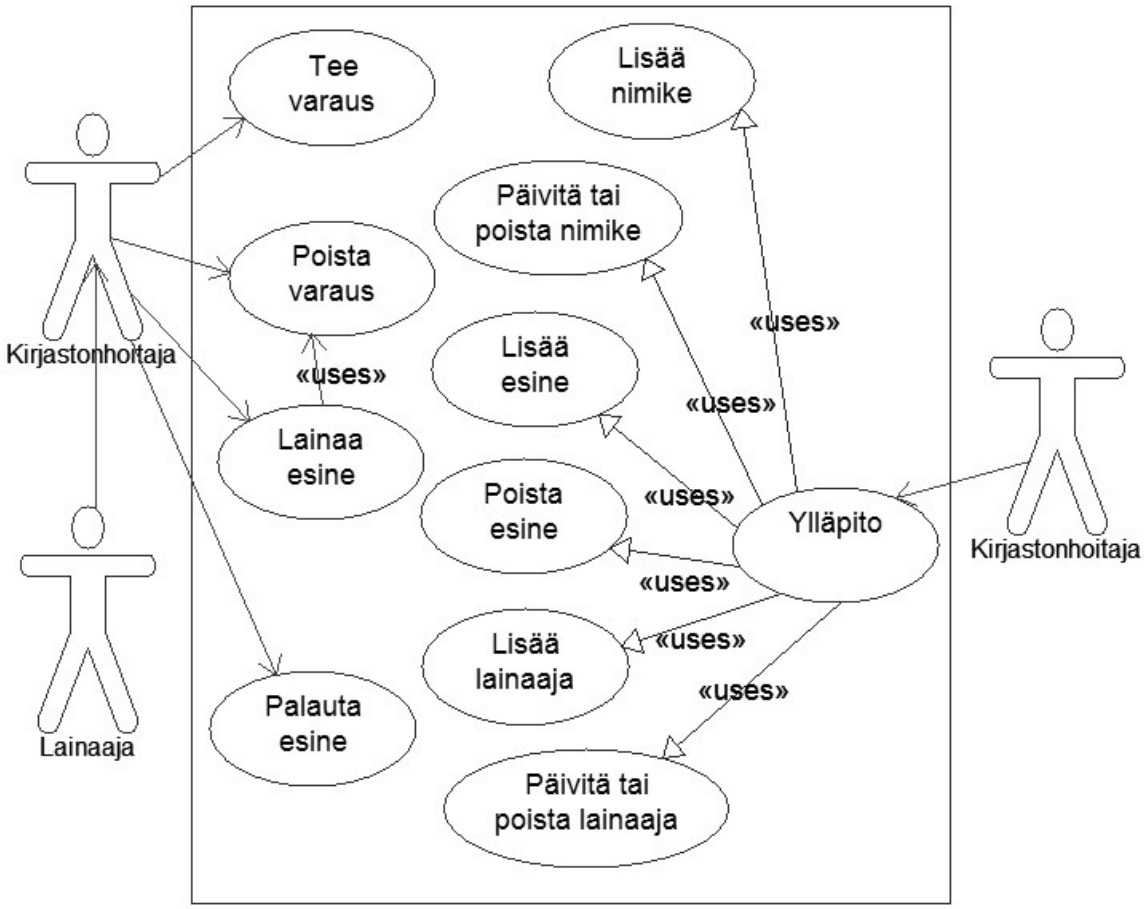
Tilalla voi olla **alitiloja** (engl. *substate*), jotka kuvaavat sen sisäistä toimintaa. Alitiloja sisältävää tilaa kutsutaan **ylitilaksi** (engl. *super state*). Alitilat voivat olla samanaikaisesti suoritettavia, jolloin niillä voidaan mallintaa rinnakkaisia säikeitä.

Siirtymä (engl. *transition*) ilmaisee siirtymistä tilasta toiseen. Siirtymään voidaan liittää varmuusehto ja toimintolausekkeita. **Varmuusehto** (engl. *guard condition*) on totuusarvolauseke, jonka on oltava tosi ennen kuin siirtymä laukaistaan. **Toimintolauseke** (engl. *action expression)* on siirtymän lauetessa suoritettavan toiminnon ilmaiseva lauseke. Siirty-

mään voi liittyä useita toimintolauseita, jotka suoritetaan järjestyksessä vasemmalta oikealle. Kuvassa 14 toimintolausekkeet ilmaisevat varausten määrän muutoksia siirtymien yhteydessä. Varmuusehdot ilmaisevat, että `Varattu`-tilasta palataan `Ei varausta` -tilaan, jos varaus poistetaan varausten määrän ollessa `1`.

3.3.6 Käyttötapauskaavio

Standardin [OMG 2011b, s. 597] mukaan **käyttötapauskaavion** (engl. *use case diagram*) avulla kuvataan tarkasteltavan järjestelmän **käyttötapaukset** (engl. *use case*), niihin liittyvät **toimijat** (engl. *actor*) ja näiden suhteet. Käyttötapausten avulla voidaan esittää järjestelmän toiminnalliset vaatimukset. Kukin käyttötapaus kuvaa järjestelmän ja sitä käyttävän toimijan välistä vuorovaikutusta tietyn tuloksen saavuttamiseksi. Kaaviossa järjestelmä esitetään laatikkona ja käyttötapaus ellipsinä kuvan 15 tapaan.



Kuva 15: Luvun 6.4 esimerkkijärjestelmän käyttötapauskaavio Rhapsody Modeler -ohjelmistolla piirrettynä.

Toimija voi olla ihminen tai toinen järjestelmä, joka käyttää kuvattavan järjestelmän palveluita. Toimijaa kuvaa kaaviossa tikku-ukkoa muistuttava näkymäelementti. Sama fyysinen käyttäjä voi esiintyä mallissa useampana kuin yhtenä toimijana. Fowler [Fowler, s. 100] esittääkin, että englanninkielisen termin *actor* sijasta termi **rooli** (engl. *role*) olisi kuvaavampi.

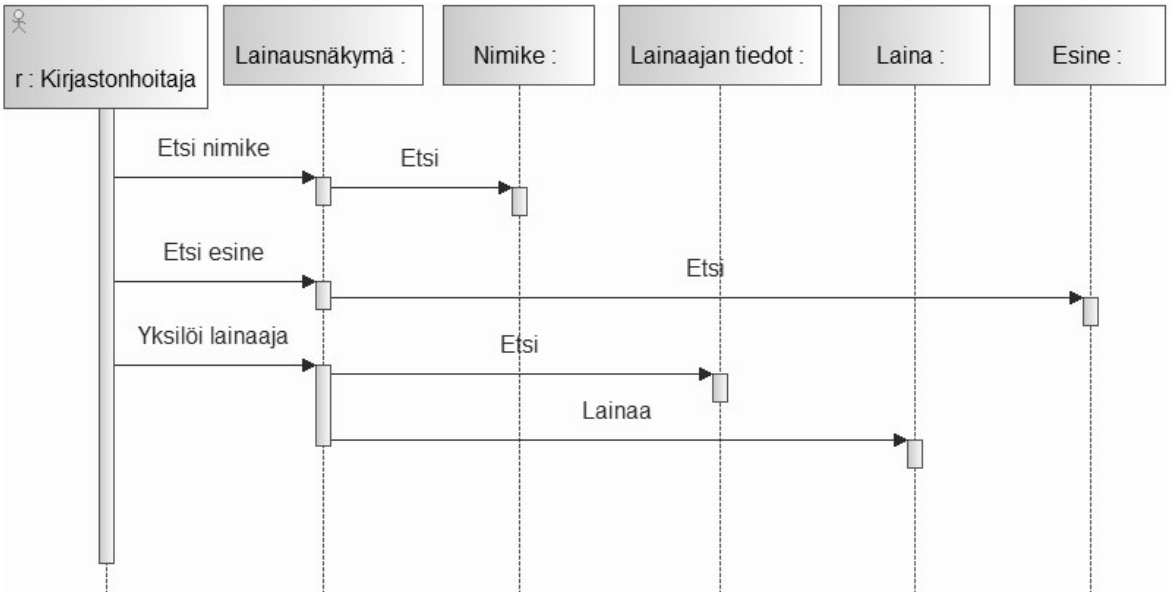
Toimijat ja käyttötapaukset yhdistetään kaaviossa toisiinsa viestiassosiaatiota kuvaavilla viivoilla. Käyttötapausten välillä voi olla erilaisia suhteita. Suhde esitetään kuvan tapaan onttopäisellä nuolella, johon liitetty stereotyyppi ilmaisee suhteen tyypin.

UML-standardi ei millään tavoin määritä tapaa, jolla käyttötapausten sisältö tulisi kuvata. Koska käyttötapausten keskeisin anti on niiden sisällön kuvaus, pelkästä kaaviosta saatava hyöty on varsin rajoitettu. Käyttötapausten toiminta kuvataan toiminnallisten näkymien avulla. Usein toiminnan kuvaamiseen käytetään luvuissa 3.3.7 ja 3.3.8 kuvattavia vuorovaikutuskaavioita.

3.3.7 Sekvenssikaavio

Standardin [OMG 2011b, s. 515] mukaan yleisin vuorovaikutusta kuvaavista kaaviotyypeistä on **sekvenssikaavio** (engl. *sequence diagram*), joka tunnetaan myös nimellä **viestiyhteyskaavio**. Se kuvaa vuorovaikutusta keskittyen vuorovaikutuksen osapuolien välillä ajan kuluessa kulkeviin viesteihin. Useimmista muista kaaviotyypeistä poiketen sekvenssikaavio ei ole verkkomainen esitys, vaan osallistujat kuvataan aikaulottuvuuden suuntaisina viivoina. UML 2 -versioissa sekvenssikaavion ilmaisuvoimaa on parannettu lisäämällä siihen esimerkiksi rakenteiset haarautumisilmaisut ja silmukkarakenne.

Sekvenssikaavioita käytetään usein, kun käyttötapauksiin halutaan liittää lisäinformaatiota. Tällöin sekvenssikaavio kuvaa, kuinka käyttötapaukseen liittyvät toimijat ovat vuorovaikutuksessa järjestelmän kanssa. Kuvassa 16 on esitetty esimerkkijärjestelmän käyttötapaukseen `Lainaa esine` liittyvä sekvenssikaavio.



Kuva 16: Esimerkkijärjestelmän käyttötapausta *Lainaa esine* kuvaava sekvenssikaavio Modelio-ohjelmistolla piirrettynä.

Osallistujat (engl. *participants*) olivat kirjan [Fowler, s. 53–54] mukaan UML 1 -versioissa olioita, mutta UML 2 -versioissa niiden tarkoitus on laajempi. Tästä syystä osallistujaa kuvaavan suorakaiteen muotoisen näkymäelementin nimi oli UML 1 -versioissa alleviivattu, mutta UML versiosta 2.0 alkaen alleviivausta ei ole käytetty.

Osallistujan olemassaoloa kuvataan aikaulottuvuuden suuntaisena viivana, jota kutsutaan **elämänviivaksi** (engl. *lifeline*). Vuorovaikutuksen jakso, jonka aikana osallistuja on aktiivinen, voidaan kuvata piirtämällä elämänviivaan palkki.

Standardin [OMG 2011b, s. 507] mukaan osallistujien väliset **viestit** (engl. *message*) esitetään nuolina kuvan 16 tapaan. Viestin nimen perässä voidaan sulkeissa esittää viestiin kuuluvat parametrit. Kirjan [Fowler, s. 55] mukaan yleensä riittää piirtää vuorovaikutuksen aloittava viesti, mutta tarvittaessa paluuviesti voidaan esittää katkoviivalla. Osallistujan omaan elämänviivaan palaavaa viestiä kutsutaan rekursioksi.

Viestin synkronisuus voidaan ilmaista käyttämällä nuolta, jonka kärki on täytetty, ja asynkronisuus käyttäen avointa nuolenkärkeä. Fowler huomauttaa kirjassaan [Fowler, s. 61],

että ennen UML:n versiota 1.4 viestin asynkronisuuden ilmaisemiseen käytettiin puolikasta avointa nuolenkärkeä.

Vuorovaikutuksen ensimmäinen viesti ei välttämättä ole minkään kaaviossa kuvatun osapuolen lähettämä, joten sen esittämiseen voidaan standardin [OMG 2011b, s. 507] mukaan tarvittaessa käyttää **löydettyä viestiä** (engl. *found message*) kuvaavaa merkintää, jossa viestiviiva alkaa täytetystä ympyrästä. Vastaavasti, mikäli viestin vastaanottajaa ei ole kuvattu kaaviossa, voidaan käyttää **kadonnutta viestiä** (engl. *lost message*) kuvaavaa merkintää, joka päättyy täytettyyn ympyrään.

Jos osallistuja ei ole olemassa koko kaavion kuvaamaa ajanjaksoa, sen luominen ja tuhoaminen on mahdollista esittää omilla merkinnöillään. Osallistujan luominen kuvataan viestillä, joka päättyy luotavan osallistujan näkymäelementtiin. Osallistujan tuhoamista kuvaa elämänviivan päähän piirretty vinoristi. Jos tuhoamisen käynnistää toiselta osallistujalta tuleva viesti, viestinuoli päättyy vinoristiin.

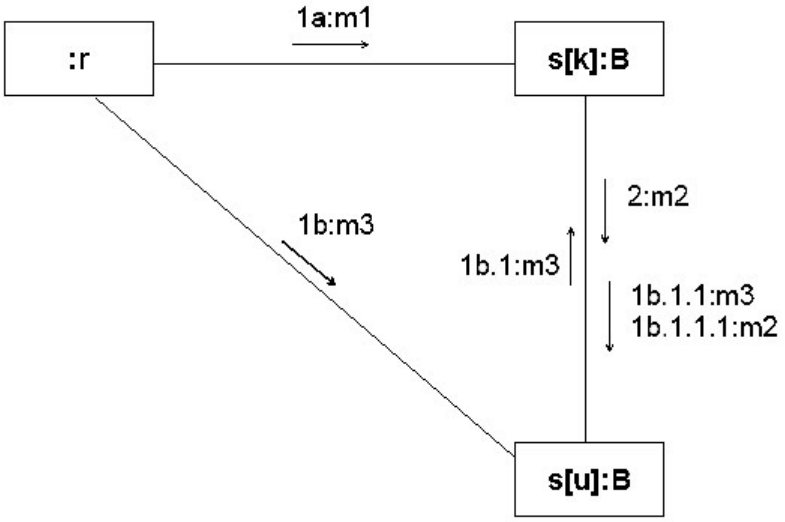
Sekvenssikaaviossa voidaan UML 2 -versiosta alkaen ilmaista silmukka- ja ehtorakenteita käyttämällä **lohkoja** (engl. *interaction frames*). Fowler tosin huomauttaa kirjassaan [Fowler, s. 57], että näiden rakenteiden esittämiseen sekvenssikaavio ei ole paras valinta. Lohkon vasemmassa yläkulmassa oleva operaattori ilmaisee, minkä tyyppisestä rakenteesta on kysymys.

3.3.8 Yhteistoiminta-, ajoitus- ja kokoava vuorovaikutuskaavio

Yhteistoimintakaavio (engl. *communication diagram*) on vuorovaikutuskaavio, joka esittää standardin [OMG 2011b, s. 524] mukaan vuorovaikutustilanteen painottaen osapuolten välisiä viestiyhteyksiä. Sekvenssikaaviosta poiketen yhteistoimintakaaviossa viestien järjestys ilmaistaan järjestysnumeroin. Yhteistoimintakaaviosta käytetään myös nimeä **kommunikointikaavio** ja UML 1 -versioissa siitä käytettiin nimeä **yhteistyökaavio** (engl. *collaboration diagram*).

Kuvassa 17 on standardin [OMG 2011b, s. 526] esimerkki yhteistoimintakaaviosta. Viestit `m1` ja `m3` lähtevät samanaikaisesti luokan `r` ilmentymästä luokan `B` kahteen ilmentymään

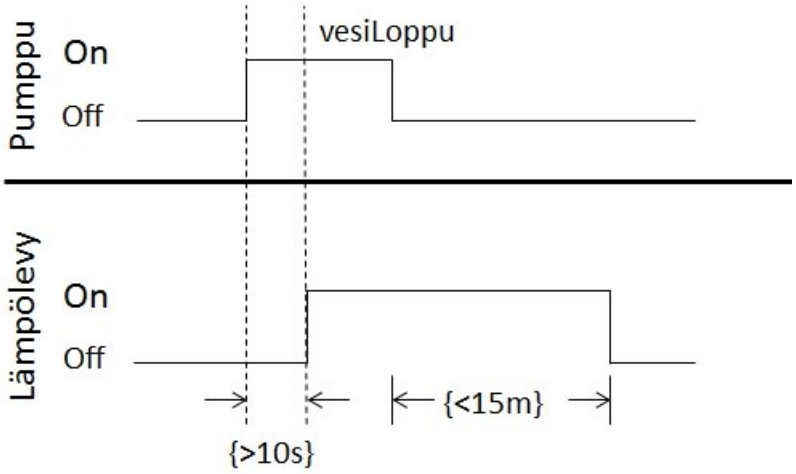
`s[k]` ja `s[u]`. UML-standardia noudattavan monitasoisen järjestysnumeroinnin mukaan esimerkissä viesti `1b.1` seuraa viestiä `1b` ja `1b.1.1` seuraa `1b.1`:tä.



Kuva 17: Yhteistoimintakaavio.

Ajoituskaavio (engl. *timing diagram*) on UML:n versioon 2.0 lisätty kaaviotyyppi. Se on standardin [OMG 2011b, s. 530] mukaan tarkoitettu käytettäväksi vuorovaikutuksen kuvaamiseen, kun halutaan esittää (reaali)ajan kuluessa tapahtuvat muutokset.

Kuvassa 18 on kirjassa [Fowler, s. 150] kuvattu esimerkki ajoituskaavion käytöstä. Kaavio esittää kahvinkeittimen lämpölevyn ja pumpun toimintaa. `Lämpölevy` kytkeytyy päälle, kun `Pumppu` on ollut toiminnassa 10 sekuntia. Veden loppuessa `Pumppu` kytkeytyy pois päältä ja `Lämpölevy` sammuu 15 minuutin kuluttua.



Kuva 18: Ajoituskaavio.

Kokoava vuorovaikutuskaavio (engl. *interaction overview diagram*) on standardin [OMG 2011b, s. 527] mukaan aktiviteettikaavion muunnos, jossa erilliset vuorovaikutuskaaviot yhdistetään toisiinsa luvussa 3.3.4 kuvatun aktiviteettikaavion rakentein.

3.4 UML:n hyödyntäminen ohjelmistojen kehittämisessä

Unified Modeling Language (UML) on yleiskäyttöinen graafinen mallinnuskieli, jota käytetään olioperustaisessa ohjelmistokehityksessä tietojärjestelmän osien ja rakenteen määrittämiseen, suunnitteluun, visualisointiin ja dokumentointiin. UML ei ole sidoksissa mihinkään tiettyyn suunnittelumenetelmään, vaan sitä voidaan hyödyntää erilaisiin suunnittelumenetelmiin liittyvien ohjelmistokehitysprosessien yhteydessä. UML-standardi ei ohjaa tapaa, jolla UML:a tulisi hyödyntää, joten käytännössä hyödyntämistä eri tilanteissa ohjaavat sovitut tai vakiintuneet toimintatavat.

3.4.1 UML:n suhde ohjelmistokehitysprosesseihin

Mallinnuskielet eivät välttämättä ota kantaa ohjelmistojen kehittämisessä käytettävään menetelmään tai prosessiin, vaan valittu suunnittelumenetelmä ja sovitut käytänteet määrittävät, kuinka mallinnuskieliä sovelletaan. Usein suunnittelumenetelmä kuvaa prosessin, jonka eri vaiheissa mallinnuskieliä käytetään sen määrittämällä tavalla.

UML:a voidaan standardin [OMG 2011a, s. 9] mukaan hyödyntää useimpien olemassa olevien olio-suuntautuneiden ohjelmistokehitysprosessimallien kanssa. UML on siis sidoksissa olioparadigmaan, mutta ei mihinkään tiettyyn suunnittelumenetelmään.

Vaikka **UML on riippumaton käytettävästä prosessimallista**, vaikuttaa valittu suunnittelumenetelmä UML:n käyttötapaan. UML:a kehitettäessä sitä on ajateltu käytettävän käyttötapauksiin pohjautuvan inkrementaalisen suunnittelumenetelmän yhteydessä, mutta kyselytutkimukseen [Dobing, s. 5] vastanneista kehittäjistä vain 44 prosenttia kertoi käyttötapauskuvauksia hyödynnettävän vähintään kahdessa projektissa kolmesta.

Kaikkiin tarkoituksiin sopivaa ideaalista ohjelmistokehitysprosessia tuskin onkaan mahdollista kehittää. Niissä esiintyy kuitenkin toisiaan vastaavia toimintoja. Kirjassa [Sommerville, s. 74] esitetään neljä ohjelmistokehitysprosessimalleille yhteistä tehtäväkokonaisuutta:

1. ohjelmiston määrittely,
2. ohjelmiston suunnittelu,
3. ohjelmiston toteutus ja validointi, sekä
4. ohjelmiston evoluutio.

Luvuissa 3.4.2–3.4.6 esitellään yleisesti UML:n kaaviotyyppien hyödyntämistä näissä ohjelmistokehitysprosessimallien tehtäväkokonaisuuksissa. Käytännössä ohjelmistokehityksessä hyödynnettävät kaaviotyypit vaihtelevat riippuen kulloisistakin tarpeista.

3.4.2 UML:n hyödyntäminen määrittelyssä

Ohjelmiston määrittelyyn sisältyvät kirjan [Sommerville, s. 75] mukaan esitutkimus, vaatimusten kartoitus ja analysointi, vaatimusten täsmentäminen sekä vaatimusten validointi. Määrittelyn tavoitteena on saavuttaa selkeä käsitys siitä, mitä järjestelmää ollaan toteuttamassa.

Esitutkimuksessa (engl. *feasibility study*) selvitetään, kannattaako järjestelmää ylipäätään toteuttaa. Tutkimukseen ei liity erityistä olionäkökulmaa, eikä UML-kaaviotyyppiä. Esitutkimukseen tosin kuuluu alustavan määrittelyn laatiminen, jossa yhteydessä UML:a voidaan hyödyntää esimerkiksi tuettavien liiketoimintaprosessien kuvaamiseen.

Vaatimusten kartoituksen ja analyysin (engl. *requirements elicitation and analysis*) aikana kartoitetaan eri sidosryhmien järjestelmälle esittämät vaatimukset, mahdollisesti mallinnetaan järjestelmää vaatimusten perusteella ja johdetaan malleista uusia vaatimuksia. **Vaatimusten täsmennys** (engl. *requirements specification*) on työvaihe, jossa kerätty informaatio muunnetaan vaatimuslistoiksi ja määrittelykuvauksiksi. **Vaatimusten validoinnin** (engl. *requirements validation*) tehtävä on varmistaa, että kuvatut vaatimukset todella määrittelevät asiakkaan haluaman järjestelmän.

UML:a hyödynnettäessä määrittelyn tuloksena on joukko luvussa 3.3.6 esiteltyjen käyttötapauksien kuvauksia ja tietojärjestelmän käsitemalli luvun 3.3.1 luokkakaavioina. Käyttötapausten pohjalta voidaan laatia luvussa 3.3.7 kuvattuja sekvenssikaavioita, joiden perusteella saadaan täsmennettyä tietojärjestelmältä odotettavat yksittäiset palvelut.

3.4.3 UML:n hyödyntäminen suunnittelussa

Ohjelmiston suunnittelussa määrittelyn tuloksena saatua mallia laajennetaan ja sitä muutetaan vastaamaan toteutuksessa käytettävän teknisen alustan vaatimuksia. Määrittelyssä laadittua luokkakaaviota täydennetään toteutukseen liittyvillä luokilla. Suunnitteluun sisältyvät kirjan [Sommerville, s. 77] mukaan arkkitehtuurisuunnittelu, abstrakti määrittely, sekä rajapinta-, komponentti-, tietorakenne- ja algoritmisuunnittelu. Kukin näistä täsmentää edellisen tuloksia ja lopputuloksena saadaan toteutettavien algoritmien ja tietorakenteiden yksityiskohtainen määrittely.

Arkkitehtuurisuunnittelussa (engl. *architectural design*) tunnistetaan ja dokumentoidaan järjestelmään toteutettavat itsenäiset kokonaisuudet, eli osajärjestelmät, ja niiden väliset suhteet. Arkkitehtuurisuunnittelua varten on kehitetty erityisesti sitä tukevia kuvauskieliä, mutta arkkitehtuuria voidaan kuvata myös lukujen 3.3.1–3.3.3 rakennekaavioiden avulla.

Abstraktin määrittelyn (engl. *abstract specification*) tuloksena saadaan arkkitehtuurisuunnittelussa löydettyjen alijärjestelmien palvelut ja toiminnan rajat määrittelevä kuvaus. Alijärjestelmä voidaan UML-kaaviossa esittää pakkausmallinnuselementillä ja alijärjestelmien suhteet kuvata luvussa 3.3.3 esiteltyä pakkauskaaviota käyttäen.

Rajapintasuunnittelussa (engl. *interface design*) suunnitellaan ja dokumentoidaan kunkin alijärjestelmän muille alijärjestelmille tarjoamat rajapinnat. Alijärjestelmän rajapinnan tulee olla sellainen, että alijärjestelmää voidaan käyttää ilman tietoa sen sisäisestä toteutuksesta. Dokumentoinnissa voidaan käyttää luvun 3.3.3 komponenttikaaviota.

Komponenttisuunnittelussa (engl. *component design*) alijärjestelmän palvelut ryhmitellään palvelut toteuttaviin komponentteihin ja komponenttien rajapinnat kuvataan. Komponenttien ja niiden rajapintojen esittämistä varten UML:ssa on käytössä luvun 3.3.3 komponenttikaavio.

Kun järjestelmä on jaettu osajärjestelmiin ja edelleen komponentteihin, **tietorakennesuunnittelussa** (engl. *data structure design*) järjestelmän toteutuksessa tarvittavat tietorakenteet suunnitellaan ja kuvataan yksityiskohtaisesti. Kuvaus voidaan esittää luvun 3.3.1 luokkakaaviona, jota havainnollistetaan tarvittaessa luvussa 3.3.2 esitellyn oliokaavion avulla.

Algoritmisuunnittelun (engl. *algorithm design*) aikana suunnitellaan ja kuvataan yksityiskohtaisesti ne palveluiden toteuttamisessa tarvittavat algoritmit, joiden toiminta ei riittävästi selviä muun laaditun kuvauksen perusteella. Algoritmien kuvaamiseen voidaan käyttää tarpeen mukaan useaa UML-kaaviotyyppiä, kuten luvussa 3.3.4 esiteltyä aktiviteettikaaviota tai luvun 3.3.7 sekvenssikaaviota.

3.4.4 UML:n hyödyntäminen ohjelmiston toteutuksessa

Toteutuksessa suunnittelun tuloksena saadut mallit ohjelmoidaan valitulla ohjelmointikielellä, sekä tehdään malliin käytännön toteutuksen vaatimat lisäykset ja ohjelmointiympäristön mahdollisesti vaatimat täsmennykset. Esimerkiksi luvun 3.3.1 luokkakaaviossa kuvattava luokkien välistä yhteenkuuluvuutta ilmaiseva koostesuhde toteutetaan eri tavoin eri ohjelmointikielissä.

Suunnittelun tuloksena saatua mallia voidaan hyödyntää ohjelmakoodin kirjoittamisen pohjana, jos käytettävä mallinnusväline tukee luvussa 5.2.3 kuvattavaa koodin generointia. Vastaavasti mallia voidaan päivittää ohjelmointityön edetessä takaisinmallinnuksen avulla.

3.4.5 UML:n hyödyntäminen verifioinnissa ja validoinnissa

Kirjassa [Sommerville, s. 80] todetaan, että ohjelmistoa verifioitaessa ja validoitaessa kehitettyä järjestelmää arvioidaan yleensä yksikkö-, integrointi-, järjestelmä- ja hyväksymistesteillä. Eri tasoilla käytetään testauksen pohjana UML-mallin eri kaavioita. Yksikkötesteissä testauksen pohjana ovat luvun 3.3.1 luokkakaaviot ja -määritykset, integrointitesteissä tavallisesti luvun 3.3.3 komponenttikaaviot ja luvun 3.3.8 yhteistoimintakaaviot sekä järjestelmätesteissä luvun 3.3.6 käyttötapauskaaviot, joita voidaan hyödyntää myös hyväksymistestauksessa. Hyväksymistestauksen tukena voidaan lisäksi käyttää esimerkiksi luvun 3.3.4 aktiviteettikaavioina tai luvun 3.3.5 tilakaavioina esitettyjä liiketoimintaprosessien kuvauksia.

3.4.6 UML:n hyödyntäminen ohjelmiston evoluutiossa

Toteutuksen jälkeen ohjelmistoa on ylläpidettävä sen elinkaaren ajan. Kirjassa [Sommerville, s. 82] käytetään termiä **evoluutio** ylläpito-toiminnon sijasta, koska entistä useammin ylläpito liittyy saumattomasti järjestelmän kehittämiseen, eikä tiukka erottelu näiden välillä vastaa todellisuutta.

UML:n näkökulmasta siirtyminen perinteisestä ylläpidosta kohti jatkuvaa evoluutiota on haaste dokumentaationa käytettyjen mallien ylläpidolle. Kirjassa [Fowler, s. 31–32] esitetään, että jokaista sovelluksen yksityiskohtaa ei kannata mallintaa dokumentaatioon. Sen sijaan mallin avulla voidaan havainnollistaa suunnittelun kannalta järjestelmän tärkeimpiä osia. Esimerkiksi luvun 3.3.3 pakkauskaavion avulla voidaan dokumentaatiossa antaa yleiskuva järjestelmän jakaantumisesta alijärjestelmiin.

UML-mallinnusta voidaan käyttää apuna myös selvitettäessä jälkikäteen ylläpidettävän järjestelmän rakennetta. Järjestelmän keskeisistä osista voidaan joko laatia UML-malli rakennetta tutkittaessa tai malli voidaan generoida automaattisesti ohjelmakoodista jotain työkalua käyttäen.

4 Käytettävyys

Sovellusprojekti-opintojaksolla ja mikroyrityksissä käytettävyyteen liittyvät piirteet ovat tärkeitä mallinnustyökalua valittaessa, koska työkalun käytön opetteluun on käytettävissä rajallisesti resursseja. Käytettävyydessä on olennaisesti kysymys siitä, kykeneekö sovelluksen käyttäjä suoriutumaan tehtävistään tuloksellisesti, tehokkaasti ja miellyttävästi. Luvussa 4.1 käsitellään käytettävyyden määritelmää ja ulottuvuuksia. Luvussa 4.2 esitellään käytettävyyden arviointia ja luvussa 4.3 arvioinnin tulosten tarkastelua.

4.1 Käytettävyyden määritelmät ja ulottuvuudet

Käytettävyydestä on esitetty useita määritelmiä. Esimerkiksi ISO-standardi 9241-11 määrittelee käytettävyyden kirjan [Barnum, s. 11] mukaan seuraavasti: *The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*. Käytettävyys on siis kokonaisuus kuvaten, missä määrin tietyt käyttäjät voivat käyttää tuotetta määrätyssä käyttötilanteessa saavuttaakseen määritetyt tavoitteet tuloksellisesti, tehokkaasti ja miellyttävästi.

Nielsen [Nielsen, s. 26] korostaa esittämässään käytettävyyden määritelmässä sitä, että käytettävyys koostuu useista ulottuvuuksista. Käytettävyyteen perinteisesti liitetyt viisi ulottuvuutta ovat opittavuus, tehokkuus, muistettavuus, virheettömyys ja tyytyväisyys. Whitney Quesenberyn 5e-määritelmän [Quesenbery, s. 2] ulottuvuuksia ovat puolestaan vaikuttava, tehokas, viehättävä, virhesietoinen ja helposti opittava.

Opittavuutta voidaan pitää kaikkein oleellisimpana käytettävyyden ulottuvuutena, sillä useimmiten järjestelmien tulee olla helposti opittavissa, jotta ne voidaan ottaa käyttöön. **Opittavuuden** (engl. *learnability*) määritelmässä Nielsen [Nielsen, s. 27] painottaa sitä, kuinka nopeasti uusi käyttäjä onnistuu perehtymään järjestelmään riittävästi työskennelläkseen sillä. Quesenbery [Quesenbery, s. 3] puolestaan huomioi myös tuen järjestelmän toiminnan perusteellisemmalle oppimiselle.

Tehokkuus (engl. *efficiency*) tarkoittaa sitä, että opittuaan käyttämään järjestelmää kokenut käyttäjä kykenee suoriutumaan tehtävistään tuloksekkaasti ja nopeasti. Tehokkuus voi

olla mitattavaa tai perustua käyttäjän tuntemukseen esimerkiksi siitä, että jokin toimenpide kestää liian kauan tai vaatii liian monta klikkausta.

Satunnaiskäyttäjät muodostavat kolmannen käyttäjäryhmän aloittelevien ja kokeneiden käyttäjien rinnalle. **Muistettavuudella** (engl. *memorability*) mitataan sitä, kuinka hyvin satunnainen käyttäjä pystyy jatkamaan järjestelmän käyttämistä oltuaan käyttämättä sitä jonkin kohtuullisen ajan. Muistettavuus on tärkeä ominaisuus esimerkiksi joillekin apuohjelmille, joita ei käytetä säännöllisesti.

Järjestelmän **virheiden määrän** (engl. *error rate*) pitäisi olla matala siten, että käyttäjä joutuu mahdollisimman harvoin virhetilanteisiin järjestelmää käyttäessään ja virheistä toipuminen on helppoa. Käytön estäviä katastrofaalisia virheitä ei saisi esiintyä lainkaan. Quesenbery [Quesenbery, s. 3] käyttää ilmaisua **virhesietoinen** (engl. *error tolerant*) painottaen sitä, kuinka hyvin järjestelmä tukee käyttäjää väistämättä esiintyvissä virhetilanteissa.

Järjestelmän käyttämisen tulisi olla miellyttävää siten, että käyttäjä kokee subjektiivisesti **tyytyväisyyttä** (engl. *satisfaction*) järjestelmää käyttäessään. Quesenbery [Quesenbery, s. 3] käyttää tässä yhteydessä ilmaisua **viehättävä** (engl. *engaging*). Poikkeavalla termivalinnalla hän on halunnut kuvata sitä, kuinka käyttöliittymä vetää käyttäjän verkkosivulle tai työtehtävän pariin, tai kuinka käyttäjä tuntee järjestelmässä käytetyn esitystavan omakseen.

Quesenberyn [Quesenbery, s. 3] mukaan **vaikuttava** (tehollinen, engl. *effective*) tuote mahdollistaa käyttäjälle järjestelmän käytölle asetetun tavoitteen saavuttamisen täysin ja täsmällisesti. Tämä on oleellista sillä, jos käyttäjä ei pysty tekemään haluamaansa asiaa, ei muilla ulottuvuuksilla ole merkitystä käyttäjän tyytyväisyydelle. Vaikuttavuuden mittaamiseksi on määritettävä, mitä tavoitteen saavuttaminen käyttäjälle tarkoittaa.

Quesenbery [Quesenbery, s. 4] huomauttaa, että käytettävyyden ulottuvuudet ovat sidoksissa toisiinsa. Esimerkiksi vaikeasti opittava tai muistettava järjestelmä on usein myös tehoton. Joskus voikin olla vaikea tulkita, mihin käytettävyyden ulottuvuuteen tehdyt havainnot liittyvät.

4.2 Heuristinen käytettävyyden arviointi

Perinteisesti käytettävyyttä on kirjan [Barnum, s. 15] mukaan arvioitu hyödyntäen muodollisia prosesseja, joissa hyödynnetään kognitiivista tutkimustietoa ja käyttöliittymäsuunnittelijoiden asiantuntemusta. Näin toteutettua arviointia varten tarvitaan tilastollisesti riittävän laaja joukko koehenkilöitä, jotka suorittavat edustavan määrän kokeita hyvin varustetussa käytettävyyslaboratoriossa.

Muodollista prosessia noudattava testaus on kallista ja aikaa vievää, minkä seurauksena käytettävyyden arvioiminen tai kehittäminen saatetaan Nielsenin [Nielsen, s. 17] mukaan jättää kokonaan tekemättä. Tästä syystä on kehitetty myös kevyempiä menetelmiä, joissa käytettävyyden arvioinnissa voidaan hyödyntää esimerkiksi käyttäjien ja tehtävien tarkkailua sekä heuristista arviointia.

Heuristisessa arvioinnissa asiantuntijat arvioivat, kuinka hyvin tarkasteltava järjestelmä noudattaa annettua sääntölistaa, eli heuristiikkaa. **Heuristiikat** koostuvat periaatteista, joiden noudattamista arvioinnissa tarkastellaan. Kirjassa [Nielsen, s. 20] kuvattu Nielsenin lista on tunnettu käyttöliittymien arviointiin käytettävä heuristiikka. Listan säännöt ovat seuraavat:

- Käyttäjän ja järjestelmän vuorovaikutuksen tulee olla yksinkertaista ja luonnollista.
- Järjestelmän tulee käyttää käyttäjän kieltä.
- Käyttäjän muistikuormitus tulee minimoida.
- Käyttöliittymän tulee olla yhdenmukainen.
- Järjestelmän tulee antaa käyttäjälle kunnollista ja riittävän nopeaa palautetta.
- Ohjelmassa ja sen eri osissa tulee olla selkeät poistumisreitit.
- Käyttäjälle tulee tarjota oikopolkuja.
- Virheilmoitusten tulee olla selkeitä ja ymmärrettäviä.
- Virhetilanteisiin joutumista tulisi välttää.
- Käyttäjän saatavilla tulee olla kunnolliset opastustoiminnot ja dokumentaatio.

Vuorovaikutuksen yksinkertaisuus tarkoittaa Nielsenin [Nielsen, s. 105] mukaan sitä, että käyttöliittymä on mahdollisimman yksinkertainen. Jokainen näytöllä oleva toiminto tai

tieto vaatii oppimista, voi aiheuttaa väärinymmärryksen ja on yksi tutkittava seikka lisää etsittäessä tehtävän suorittamisessa tarpeellisia asioita.

Käyttäjäkeskeiseen suunnitteluun kuuluu, että käyttöliittymän kieli perustuu **käyttäjän kieleen**, eikä järjestelmän teknisestä toteutuksesta lähtöisin olevaan sanastoon. Esimerkiksi koodistojen arvot tulee esittää selkokielisinä, kuten `Jyväskylä`, eikä numeroarvoina, kuten `179`. Tärkeää on myös olla käyttämättä sanoja epätavallisissa merkityksissä. Kirjoitetun tekstin ohella sääntö koskee myös käyttöliittymän graafisia symboleita.

Kirjan [Nielsen, s. 129] mukaan ihmisten on yleensä paljon vaikeampi palauttaa asioita mieleen ilman apua, kuin tunnistaa tuttuja asioita niitä nähdessään. Ohjelmiston tulisi pienentää **käyttäjän muistikuormaa** esimerkiksi valintalistojen ja valikkojen avulla.

Yhdenmukaisuus on kirjan [Nielsen, s. 132] mukaan yksi käytettävyyden perusperiaatteista. Saman toiminnon tulisi aina aiheuttaa sama lopputulos, ja sama informaatio tulisi sijoittaa kaikissa näytöissä samassa muodossa samaan kohtaan.

Järjestelmän tulisi jatkuvasti informoida käyttäjää meneillään olevasta toiminnasta ja käyttäjän syötteiden käsittelystä. **Palautteen antaminen** ei saa rajoittua virhetilanteisiin. Palaute on erityisen tärkeää, jos järjestelmä tekee pitkäkestoista operaatiota, joka pidentää vasteaikaa. Käyttäjälle tulisi pyrkiä antamaan palautetta myös järjestelmän vikatilanteissa.

Käyttäjät eivät kirjan [Nielsen, s. 138] mukaan halua tuntea joutuvansa peruuttamattomasti ansaan tietokonetta käyttäessään. Järjestelmän tulisi tarjota selkeä **poistumisreitti** aina, kun se on mahdollista. Poistumisreittinä voi toimia esimerkiksi peruutustoiminto, jolla on mahdollista palata toimenpidettä edeltävään tilaan.

Kokeneiden käyttäjien tulisi voida suorittaa usein toistuvia toimintoja **oikopolkuja** käyttäen. Oikopolkuja voivat olla esimerkiksi pikanäppäimet, ennakoiva tekstinsyöttö ja oletustoiminnon suorittaminen, kun toimenpiteen kohdetta kaksoisnapsautetaan hiirellä.

Kirjan [Nielsen, s. 142] mukaan virhetilanteet on käytettävyyden kannalta tärkeä huomioida kahdesta syystä. Ensinnäkin virhetilanteessa käyttäjä on kohdannut ongelman, eikä mahdollisesti pysty suorittamaan haluamaansa toimintoa. Toiseksi virhetilanteessa käyttäjää voidaan auttaa ymmärtämään järjestelmän toimintaa paremmin, koska tällöin hän aina-

kin jossain määrin on kiinnostunut virheilmoituksen sisällöstä ja järjestelmällä on jotain tietoa virheen syystä.

Hyvä virheilmoitus noudattaa kirjan [Nielsen, s. 142–144] mukaan neljää sääntöä:

1. Virheilmoituksessa käytettävän kielen tulee olla siinä määrin selkeää, että ilmoitus on ymmärrettävissä ilman ulkopuolisia tietolähteitä.
2. Virheilmoitusten tulee olla täsmällisiä.
3. Virheilmoitusten tulee rakentavasti auttaa käyttäjää ongelman ratkaisemisessa.
4. Virheilmoitusten tulee olla kohteliaita, eikä niissä tule syyllistää käyttäjää virheestä.

Vielä hyvää virheilmoitusta parempi on se, että **virhetilanteeseen päätyminen voidaan välttää**. Peruuttamattomien toimintojen yhteydessä hyvä keino on varmistaa, että käyttäjä todella haluaa suorittaa valitsemansa toimenpiteen. Virhetilanteiden välttämiseksi on myös hyvä tehdä järjestelmän komennoista selkeästi toisistaan eroavia.

Vaikka parhaassa tapauksessa järjestelmä on niin helppokäyttöinen, että **käyttöohjeet** ja oppaat ovat tarpeettomia, tämä ei aina ole mahdollista. Kirjassa [Nielsen, s. 149] todetaan, että käyttäjät eivät yleensä lue käyttöohjeita. Jos siis käyttäjä käyttöohjetta lukee, hänellä on jokin välitöntä vastausta vaativa ongelma. Näin ollen tietojen tulisi olla käyttöohjeessa helposti löydettävissä, ja ohjeissa tulisi olla hakutoiminto. Ohjeen tulisi myös olla tarvittaessa helposti löydettävissä.

4.3 Heuristisen arvioinnin tulosten tarkastelu

Heuristisessa arvioinnissa asiantuntijat arvioivat, kuinka hyvin tarkasteltava järjestelmä noudattaa annettua sääntölistaa. Tuloksia tarkasteltaessa tulee huomioida se, että havainnot ovat vakavuudeltaan erilaisia, sekä se, että arvioijien kokemus vaikuttaa tehtyjen havaintojen määrään.

Käytettävyysperiaatteiden vastaisuuksista tehdyt havainnot voidaan luokitella esimerkiksi Nielsenin kirjassaan [Nielsen, s. 103] esittämällä viisikohtaisella vakavuusasteikolla:

0. Kyseessä ei ole käytettävyysongelma.
1. Kosmeettisella ongelmalla on vähäinen vaikutus käytettävyyteen.

2. Pienellä ongelmalla on vaikutusta käytettävyyteen.
3. Isolla ongelmalla on merkittävää vaikutusta käytettävyyteen.
4. Katastrofaalinen ongelma on este järjestelmän käyttämiselle.

Lopputuloksena arvioinnista saadaan lista havaituista käytettävyyspuutteista, jotka on kytketty rikottuun periaatteeseen.

Nielsen [Nielsen, s. 156] arvioi yhden asiantuntijan löytävän tavallisesti noin 35 prosenttia käyttöliittymän käytettävyysongelmista. Koska eri asiantuntijat löytävät usein eri ongelmia, kasvaa löydettyjen ongelmien määrä, kun asiantuntijoiden määrää lisätään. Viisi asiantuntijaa löytää noin 75 prosenttia ongelmista, minkä jälkeen löydettyjen uusien ongelmien määrä tasaantuu.

Asiantuntijan kokemuksella on Nielsenin kirjan [Nielsen, s. 160] mukaan vaikutusta löydettyjen käytettävyysongelmien määrään. Aloittelevat käytettävyysasiantuntijat löysivät järjestetyssä kokeessa keskimäärin 22 prosenttia käytettävyysongelmista, kun taas kokeneet asiantuntijat löysivät 41 prosenttia. Kaikkein parhaan tuloksen saavuttivat arvioijat, jotka olivat sekä yleisesti käytettävyyden että arvioitavan käyttöliittymän sovellusalueen asiantuntijoita. He löysivät 60 prosenttia käytettävyysongelmista.

5 Mallinnusvälineiden vertailukohteet

Luvussa esitellään lähteissä [Auer], [Eichelberger], [Eriksson], [Khaled], [Kirchner], [Koskimies] ja [Rumbaugh] esitettyjä UML-mallinnusvälineiden ominaisuuksia ja vertailukohteita, arvioidaan niiden soveltuvuutta käytettäväksi tutkielmassa ja esitetään perustelu kunkin vertailukohdan valinnalle tai rajaamiselle vertailun ulkopuolelle. Jotta mallinnusvälineiden arvioinnista ja vertailusta olisi saatavissa toivottu hyöty, on valittava tutkielman tavoitteiden kannalta oleelliset vertailukohteet ja niiden tarkastelutavat.

Muutamia vertailukohteita on lähteissä käsitelty eri aihepiirien alla. Tällaisten vertailukohtien käsittely on pyritty sijoittamaan luvussa siihen alalukuun, johon se tarkasteltavalla kohdealueella tuntuu luontevimmin sopivan.

Luvussa 5.1 kuvataan vertailukohteiden valinnan perusteluita. Luvussa 5.2 esitellään UML-mallinnukseen liittyvien vertailukohteiden valintaa ja käsittelyä tutkielmassa. Luku 5.3 käsittelee käytettävyyteen liittyvien vertailukohteiden tarkasteluun liittyviä valintoja, sekä luku 5.4 elinkaarikustannusten arviointia.

5.1 Vertailukohteiden valinta

Vertailukohteet ovat luvun 2.4 tavoitteiden pohjalta jaettavissa kolmeen ryhmään, jotka ovat tuki UML-mallin laatimiselle, käytettävyys ja elinkaarikustannukset. UML:en ja mallintamiseen liittyvät vertailukohteet esitellään luvussa 5.2. Käytettävyyttä arvioidaan ensisijaisesti luvussa 4.2 esitellyllä heuristisella menetelmällä, jonka lisäksi arvioinnin tukena käytetään luvussa 5.3 esiteltäviä vertailukohteita. Elinkaarikustannusten arviointiin liittyvät vertailukohteet esitellään luvussa 5.4.

Ohjelmistojen arvioitavat ominaisuudet valittiin aluksi kirjallisuuden perusteella, minkä jälkeen niitä sovitettiin ja tarkennettiin tutkielman tavoitteisiin sopiviksi. Näin luotuja vertailukohteita sovellettiin kolmeen avoimen lähdekoodin UML-mallinnusvälineeseen. Lisäksi samoja vertailukohteita soveltaen tarkasteltiin soveltuvin osin vertailukohtana olevaa suljetun lähdekoodin ohjelmistoa.

Artikkelissa [Eichelberger, s. 49] todetaan monimutkaisten mallinnustyökalujen vertailun toimintotasolla olevan valtaisa työ, joten vaatimukset ja arvosteluperusteet oli sovitettava tutkielman tavoitteisiin sopiviksi vertailukohteiksi. Myös valittavien vertailukohteiden määrää oli rajoitettava.

Vertailukohteiden valintaa ja painotusta ohjasivat Sovellusprojektien tarpeet ja mikroyritysten tarpeista esitetyt oletukset. Vertailukohteiden oli oltava sellaisia, että niiden avulla ohjelmistojen arviointi oli mahdollista tehdä tutkielman toteutuksen kannalta mielekkäässä ajassa ja käytettävissä olevin resurssein. Käytettävissä oleva aika ja muut resurssit oli huomioitava erityisesti käytettävyyden arvioinnissa. Myös tutkielman tavoitteet huomioiden oli järkevää hakea sellaisia vertailukohteita, joiden tarkasteleminen oli tehtävissä myös muille vastaaville UML-mallinnusvälineille rajoitetussa ajassa ja rajatuin resurssein.

Luvuissa käsiteltyjen vertailukohtien lisäksi edellä mainituissa lähteissä esitetään muutamia sellaisia mallinnusvälineiltä toivottavia ominaisuuksia, jotka eivät kuulu tutkielman piiriin. Tutkielman arvioinnissa ei tarkasteltu integraatioita muihin välineisiin, tukea ohjelmistoprosesseille, eikä tukea yleisille ohjelmistoratkaisuille, kuten suunnittelumalleille. Niiden osalta Sovellusprojektien tarpeet poikkeavat merkittävästi toisistaan.

5.2 UML-mallinnukseen liittyvät vertailukohteet

Perusvaatimus UML-mallinnusvälineelle on, että sillä on mahdollista laatia UML-kaavioita. UML-kaavioita voi piirtää piirto-ohjelmistoilla tai paperia ja kynää käyttäen. Näistä poiketen mallinnusväline mahdollistaa mallin tallentamiseen mallikantaan ja siirron muihin mallinnusvälineisiin.

Luvuissa 5.2.1–5.2.5 käsitellään niitä lähteissä [Auer], [Eichelberger], [Eriksson], [Fowler], [Khaled], [Kirchner] ja [Rumbaugh] esitettyjä vertailukohteita, jotka liittyvät tietojärjestelmien mallintamiseen UML:a käyttäen. Luvussa 5.2.6 on yhteenveto vertailuun valituista vertailukohteista.

5.2.1 Kattava UML-tuki

Artikkelissa [Khaled, s. 112] todetaan, että UML-mallinnusvälineen tulisi tukea kaikkia UML-kaavioita, minkä katsotaan tarkoittavan UML:n version 1.5 osalta tukea yhdeksälle ja UML 2 -versioiden osalta yhdelletoista kaaviotyypille. Aivan kaikkien standardissa määriteltyjen kaaviotyyppien tukea ei siis kuitenkaan edellytetä. Artikkelissa [Eichelberger, s. 48] esitellyssä arvioinnissa on selvitetty muun muassa välineiden kaaviotukea ja todetaan, että vielä vuonna 2009 vain muutamat välineet tukivat UML 2.0 -versioon lisättyjä kaaviotyyppejä. Vuonna 2008 julkaistun kyselytutkimuksen [Dobing, s. 5 ja 6] mukaan ohjelmistokehityksessä käytetyimpiä ovat luokka-, käyttötapaus- ja sekvenssikaaviotyypit, sekä yhdessä projektissa käytetään yleisimmin 3–4 kaaviotyyppiä.

UML-tuen arvioimista **tuettavien ominaisuuksien** tarkkuudella käsitellään laajimmin artikkelissa [Kirchner, s. 16–18]. Vaikka vuonna 2001 julkaistussa artikkelissa arvioinnin kohteena on UML:n versio 1.3, voitaneen yleisiä periaatteita hyödyntää arvioitaessa yhteensopivuutta muidenkin versioiden kanssa. Yksityiskohtaisemmassa arvioinnissa tuetulla UML-versiolla on kuitenkin merkitystä. Esimerkiksi Fowler kuvaa kirjassaan [Fowler, s. 57–61] vuodelta 2004, kuinka sekvenssikaavioissa käytetty silmukkarakenteen esitystapa on vaihdellut merkittävästi eri UML-versioissa.

Esimerkiksi luokkakaavion osalta Kirchner ehdottaa artikkelissaan [Kirchner, s. 16–18] tarkastettavaksi, ovatko abstraktiluokka, komponenttiluokka, ainokainen (engl. *singleton*), assosiaatioluokka, metaluokka, toteutusluokka, luokkakategoria ja rajapinta esitettävissä. Vuorovaikutusta kuvaavien kaavioiden osalta puolestaan ensimmäisenä selvitettäväksi asiaksi nostetaan tuki sekvenssi- ja yhteistyökaaviolle, joista jälkimäinen on uudelleennimetty UML 2.0 -versiosta alkaen yhteistoimintakaavioksi.

Tutkielmassa on Kirchnerin artikkelissaan [Kirchner, s. 16–18] ehdottamien tarkastelukohteiden sijasta selvitetty mahdollisuutta kirjassa [Fowler] kuvattujen merkintätapojen käyttämiseen. Tähän on päädytty, koska Kirchner viittaa vanhempaan UML-versioon 1.3 ja tarkasteltavia kaaviotyyppejä koskevia tarkastelukohteita on luokkakaaviota lukuunottamatta artikkelissa vain muutamia.

Kirjassa [Fowler, s. 14–16] huomautetaan, että joissain tapauksissa on hyödyllistä liittää malliin myös **UML-standardiin kuulumattomia kaavioita**. Mallinnusvälineet saattavatkin sisältää tämän tapaisia laajennoksia. Ne voivat olla lisäyksiä UML-kaavioihin, kuten silmukoiden esittämisen mahdollistaminen sekvenssikaaviossa, tai kokonaisia kaaviotyyppejä, kuten ERM-kaavio.

Näiden laajennosten valikoima vaihtelee välineittäin, joten niiden osalta yhtenäinen arviointi on vaikeaa. Mahdolliset UML-standardiin kuulumattomat kaaviotyypit mainitaan välineiden esittelyn yhteydessä luvuissa 6.2.1–6.2.4.

5.2.2 Mallikannan hallinta

Malli ja kaavioihin liittyvä visuaalinen informaatio tallennetaan **mallikantaan**. Kirjassa [Eriksson, s. 30] todetaan, että mallikannassa tulisi olla koko mallin yleiset tiedot, joita tarkastellaan eri kaavioiden kautta.

Mallikantaan tukeutuvia palveluja ovat kirjan [Eriksson, s. 32] mukaan mallin ristiriitojen tarkistaminen, kritisointi ja raportointi, sekä elementtien ja kaavioiden uudelleenkäyttö. Kirjassa [Rumbaugh, s. 109] mainitaan lisäksi mallin versiointi, pääsynhallinta ja konfiguraation hallinta. Artikkelissa [Kirchner, s. 14] huomautetaan, että vaikka väline ei tukeutuisikaan mallikantaan, saattaa osa näistä toiminnoista olla toteutettuna muilla keinoin. Mallikantaan tukeutuvat ominaisuudet esitellään kunkin ohjelmiston arvioinnin yhteydessä luvussa 6.2.

Ristiriitojen tarkistaminen tarkoittaa sitä, että väline ilmaisee tai estää muutoksen, joka johtaisi mallinnuselementin ristiriitaiseen käyttöön eri kaavioissa. Käyttäjää tulee esimerkiksi varoittaa, jos hän yrittää poistaa useassa kaaviossa käytetyn elementin vain yhdestä kaaviosta. Artikkelissa [Kirchner, s. 14] todetaan, että ristiriitojen tarkastaminen voi tarkoittaa yksinkertaisimmillaan nimiavaruuden tietotyyppien vastaavuuksien tarkistamista ja monipuolisimmillaan kokonaisten samoja elementtejä käyttävien kaavioiden yhdenmukaisuuden tarkastamista.

Kritisoinnilla tarkoitetaan toimintoa, jossa väline analysoi mallin tietoja osoittaen määrittelemättä jääneitä kohtia, sekä etsii virheitä tai sopimattomia ratkaisuja. **Kaavioiden ja**

elementtien uudelleenkäyttö mahdollistaa laadittujen kaavioiden ja mallinnuselementtien käyttämisen uusissa malleissa.

Artikkelissa [Kirchner, s. 14] todetaan, että välineet tarjoavat **versiohallintaan** liittyviä toimintoja joko välineeseen itseensä toteutettuina tai ulkoista versiohallintaa hyödyntäen. Molemmissa tapauksissa versiohallinnan toiminnallisuus vaihtelee versioitavissa olevien kokonaisuuksien koon ja eri käyttäjien kehittämien versioiden yhdistämismahdollisuuksien suhteen.

Pääsynhallinnan avulla rajoitetaan eri käyttäjien oikeuksia käsitellä mallia. Osalle mallinnusvälineen käyttäjistä voidaan esimerkiksi myöntää vain lukuoikeus, jolloin he eivät voi tehdä muutoksia malliin, mutta pystyvät kuitenkin tarkastelemaan sitä halutessaan.

Kirjassa [Eriksson, s. 33] esitetään, että työvälineen tulisi mahdollistaa usean käyttäjän yhtäaikainen työskentely saman mallin parissa. Artikkelin [Koskimies, s. 40] mukaan tällöin on huolehdittava siitä, että käsittely tapahtuu kontrolloidusti. Kirjassa [Eriksson, s. 33] esitetään keinoksi esimerkiksi sitä, että käyttäjä lukitsee käsittelemänsä kaavion, jolloin muiden siihen tekemät samanaikaiset muutokset estyvät. Työvälineen tulisi myös havaita mallikannassa oleviin yhteisiin elementteihin kohdistuvat muutokset ja antaa käyttäjälle mahdollisuus hyväksyä tästä seuraavat muutokset.

Sommerville [Sommerville, s. 690] kuvaa **konfiguraationhallinnan** olevan kehitettävän ohjelmiston hallintaan tarkoitettujen standardien ja prosessien kehittämistä ja hyödyntämistä. Konfiguraationhallintavälineitä käytetään mallin osien eri versioiden säilyttämiseen, mallin koostamiseen näistä osista ja julkaistujen versioiden seurantaan.

Raportoinnin osalta artikkelissa [Kirchner, s. 13] esitetään, että välineen tulisi kyetä tarjoamaan käyttäjälle tilastotietoa esimerkiksi luokkien metodien määrästä, perinnän syvyydestä ja moduulien sisäisestä yhtenäisyydestä (engl. *cohesion*). Tätä informaatiota voidaan käyttää apuna mallin monimutkaisuutta ja laatua arvioitaessa.

5.2.3 Koodin generointi ja takaisinmallinnus

Koodin generointi tarkoittaa toimintoa, jonka avulla mallin pohjalta generoidaan määrätylle ohjelmointikielelle ohjelmakoodia. Monet työkalut mahdollistavat artikkelin [Koskimies, s. 38] mukaan suoritettavan koodin generoimisen mallista eri kohdekieliin, vaikka toiminnolla arvellaankin olevan toistaiseksi varsin vähäinen merkitys UML:n käytölle ohjelmistokehityksessä. Koodia generoimalla voidaan kehittää esimerkiksi järjestelmän rakenteen kuvaava koodirunko luokkakaavioiden perusteella ja metodien rungot sekvenssi- tai tilakaavioiden perusteella.

Takaisinmallinnukseksi (engl. *reverse engineering*) kutsutaan kirjan [Booch, s. 16] mukaan toimintoa, jossa työkalu kehittää automaattisesti olemassa olevasta järjestelmästä osia malliin. Yleisimmin takaisinmallinnus laatii luokkakaavioita olemassa olevan lähdekoodin perusteella. On myös esimerkiksi mahdollista kehittää sekvenssikaavioita ajettavasta ohjelmasta tai luoda mallissa kuvattujen olioiden säilyvyyttä koskevien merkintöjen perusteella vastaavat tietokantataulut, kuten artikkelissa [Kirchner, s. 11] esitetään.

Sekä koodin generoinnissa että takaisinmallinnuksessa on artikkelin [Kollmann, s. 1–2 ja 8] mukaan ongelmallista se, että koodin ja kaavioiden välillä ei ole suoraa käsitteellistä vastaavuutta. Tämän seurauksena eri työkalut tuottavat takaisin mallinnettaessa samasta lähdekoodista toisistaan poikkeavia kaavioita. Lisäksi artikkelissa [Kirchner, s. 11] huomautetaan, että esimerkiksi periytymisen toteutukset vaihtelevat eri oliokielissä, mikä vaikuttaa sekä generoitaessa koodia tietyille kohdekielille että takaisinmallinnettaessa. Tämä puolestaan voi johtaa siihen, että malliin muodostuu riippuvuuksia tiettyyn toteutuksessa käytettyyn ohjelmointikieleen.

Artikkeleissa [Khaled, s. 112] ja [Kirchner, s. 10] nostetaan yhdeksi UML-mallinnusvälineiden vertailukohdaksi tuki edestakaiselle mallinnukselle (engl. *round-trip engineering*). **Edestakaisella mallinnuksella** tarkoitetaan toimintoa, jossa malli ja sitä vastaava lähdekoodi synkronoidaan keskenään koodin generoinnin ja takaisinmallinnuksen yhdistelmällä. Edestakaisen mallinnuksen osalta on huomioitava mallinnusvälineen tukemat kaaviotyypit samalla tavalla kuin koodin generoinnin ja takaisinmallinnuksen yhteydessäkin. Artikkelissa [Koskimies, s. 41] arvellaan edestakaisen mallinnuksen avulla

jatkuvasti synkronoituvan mallin soveltuvan käytettäväksi ketterien sovelluskehitysmenetelmien yhteydessä. Sitä vastoin Rumpe uskoo artikkelissaan [Rumpe, s. 4] edestakaisen mallinnuksen merkityksen pienenevän ketterissä menetelmissä samaan tapaan kuin ohjelmointikielten kääntäjien kehityksen yhteydessä tapahtui korkeamman tason ohjelmointikielten ja assembler-kielen välisille käännöksille.

Koodin generoinnin voi ajatella tukevan projektiryhmän toimintaa mahdollistamalla mallin hyödyntämisen suoraan ohjelmointityön pohjana. Esimerkiksi luokkakaavion laatiminen voi tuntua mielekkäämmältä, jos sen perusteella on mahdollista generoida osa ohjelmakoodista. Mahdollisuus koodin generoinnin hyödyntämiseen on kuitenkin käytännössä arvioitava projektikohtaisesti, koska se edellyttää työkalulta tukea valitulle ohjelmointikielelle. Yleisesti on todettu, että ohjelmistotuotannossa toiminnon merkitys on vähäinen. Generoidun koodin laatua ei tutkielmassa arvioida, mutta tuetut ohjelmointikielet huomioidaan luvussa 6.2 ohjelmistojen esittelyssä. Lisäksi mainitaan ne kaaviotyypit, joista koodin generoiminen on mahdollista.

Takaisinmallinnuksesta on hyötyä, mikäli kehitetään uusia ominaisuuksia aiemmin kehitettyyn heikosti dokumentoituun toteutukseen perustuen. Takaisinmallinnuksen hyödyntäminen edellyttää tukea lähdekoodissa käytetylle ohjelmointikielelle, joten toiminnon hyödyllisyys on arvioitava tapauskohtaisesti. Sivun [Heikkilä] mukaan Sovellusprojekti-opintojaksolla on toteutettu muutamia projekteja, joissa on jatkokehitetty olemassa olevaa sovellusta. On kuitenkin vaikeaa arvioida, olisiko takaisinmallinnuksesta ollut hyötyä näissä projekteissa. Takaisinmallinnuksen toimivuutta ei tutkielmassa erikseen kokeilla, mutta tuetut ohjelmointikielet tuodaan kuitenkin ilmi työkalujen esittelyssä luvussa 6.2. Lisäksi tuodaan esille ne kaaviotyypit, joista koodin generoiminen on mahdollista. Mahdollisuutta edestakaiseen mallinnukseen ei erikseen selvitetä.

5.2.4 Mallin dokumentointi

Artikkelin [Kirchner, s. 12–13] mukaan **dokumentoinnilla** tarkoitetaan mallinnusvälineiden yhteydessä mahdollisuutta kommentoida mallia ja sen osia sekä jakaa tämä informaatio sähköisessä muodossa tai paperilla. Mahdollisina sähköisinä esitysmuotoina mainitaan

HTML ja XML, sekä PDF-, RTF- ja Word-tiedostomuodot. Grafiikan esittämiseen sopivista tiedostomuodoista mainitaan GIF, BMP ja JPG.

Artikkeleissa [Auer] ja [Kirchner, s. 10] huomautetaan, että UML-mallia olisi tarpeellista pystyä esittelemään myös **muissa kuin mallinnusohjelmissa**. Esimerkiksi UML-mallin osia voitaisiin liittää tekstinkäsittelyasiakirjaan, PDF-dokumenttiin tai esitysgrafiikkaan. Artikkelissa [Khaled, s. 112] esitetään, että ohjelmistosta tulisi voida viedä kaavioiden esityksiä tekstinkäsittelyohjelman tai WWW-selaimen ymmärtämässä muodossa. Mahdollisina tiedostomuotoina mainitaan esimerkiksi PDF, GIF ja JPEG.

Artikkelissa [Khaled, s. 111] esitetään, että välineen tulisi pystyä muodostamaan mallista **HTML-dokumentaatio**, joka sisältää kuvan jokaisesta mallin kaaviosta ja navigoinnissa tarvittavat hyperlinkit. HTML-dokumentaatiolla voi olettaa olevan mikroyrityksissä enemmän merkitystä kuin Sovellusprojekti-opintojaksolla, koska dokumentaatio on pidettävä ajantasaisena evoluution aikana.

Mallin esittämistä paperilla ei pidetä artikkelissa [Kirchner, s. 13] toivottavana tai käytännöllisenä, mutta sen tulee kuitenkin olla tarvittaessa mahdollista. Perustoimintona nähdään riittäväksi ruudulla näkyvän mallin osan **tulostaminen** sellaisenaan, paperiarkille sovitettuna tai eri tavoin suurennettuna. Tulostuksen esikatselua pidettiin myös toivottavana ominaisuutena.

5.2.5 Mallien ja kaavioiden siirtäminen

Artikkelissa [Kirchner, s. 10] esitetään, että tulisi olla esimerkiksi tarvittaessa mahdollista siirtää kehitysympäristöön liitetyllä yksinkertaisella mallinnusvälineellä kehitetty malli monipuolisempaan välineeseen jatkokäsittelyä varten. Lisäksi saattaa olla hyödyllistä tuoda laajalti käytetyissä välineissä kehitettyjen mallien osia semantiikka säilyttäen käytettäväksi omassa ympäristössä.

Yhdellä välineellä luodun mallin siirtäminen toiseen välineeseen edellyttää muun muassa **standardia mallin tallennusmuotoa**. Standardin [OMG 2011a, s. 211] mukaan XML Metadata Interchange (XMI) on OMG:n kehittämä tiedonsiirtomuoto, joka mahdollistaa UML-mallin tallentamisen ja siirtämisen XML-muodossa.

Artikkelissa [Koskimies, s. 42] XMI:n puutteeksi nähdään se, että se ei sisällä **UML-kaavioiden asettelutiedon** esittämisessä tarpeellisia mekanismeja. Tästä syystä työvälineissä on ollut tarpeen käyttää välinekohtaisia laajennoksia asettelutiedon esittämisessä, minkä johdosta välineet tukevat XMI:tä kukin omalla tavallaan. Koska muunkin kuin malli-informaation siirtäminen on katsottu oleelliseksi, OMG julkaisi vuonna 2006 XMI:en linkittyvän kaavioiden asettelutiedon välittämiseen keskittyvän Diagram Interchange -nimisen määrityksen. Se ei kuitenkaan saavuttanut OMG:n [OMG 2010, s. 13] mukaan laajaa suosiota, joten se korvattiin uudella The Diagram Definition -nimisellä määrityksellä vuonna 2010.

5.2.6 Huomioitavat UML-mallinnukseen liittyvät vertailukohteet

Kattava UML-tuki on mallinnusvälineiden käyttötarkoituksen kannalta keskeinen tarkasteltava ominaisuus. Välineen ei kuitenkaan ole välttämätöntä tukea kaikkia UML-kaaviotyyppejä, koska Sovellusprojekteissa käytetään yleensä vain luvussa 2.1 mainittua kuutta kaaviotyyppiä. Lisäksi standardissa [OMG 2011b, s. 1] todetaan, että kaikissa toimintaympäristöissä kaikki UML-mallinnuskielen sisältämät kuvaustavat eivät välttämättä ole tarpeellisia.

Vertailussa keskitytään arvioimaan tukea yleisimmin Sovellusprojekti-opintojakson projekteissa käytetyiksi arvioiduille kaaviotyypeille, joita ovat luokka-, käyttötapaus-, sekvenssi-, aktiviteetti- ja tilakaaviot. UML-tuen tarkastelu perustuu lähteiden [Fowler], [Khaled] ja [Kirchner] pohjalta muodostettuun listaan arviointikohteista. Tarkemmat arviointikohteet on esitetty liitteessä 1. Kaikki kunkin mallinnusvälineen tukemat kaaviotyypit luetellaan luvussa 6.5.1.

Mallikantaan liittyvistä toiminnoista havainnoidaan arvioinnin yhteydessä ristiriitojen tarkastamista ja mallin kritisointia. Nämä toiminnot tukevat kokematonta käyttäjää mallia laadittaessa ja voivat auttaa kokenuttakin käyttäjää huomaamaan mallissa olevia puutteita. Vertailussa todetaan, onko ohjelmistoissa kyseisiä toimintoja.

Usean käyttäjän tuen toteutusta ei toimintotasolla kokeilla, mutta sen olemassaolo tuodaan kuvauksessa esille, mikäli asia selviää dokumentaatiosta tai muiden vertailukohtien tarkas-

telun yhteydessä. Sovellusprojekti-opintojaksolla työryhmä on pieni, jolloin todennäköisyys sattumalta tapahtuvalle mallin yhtäaikaiselle muuttamiselle on pieni. Todennäköisyyttä vähentää myös se, että yleensä mallintamisesta vastaa yksi ryhmän jäsen.

Dokumentaatiosta ilmenneet mahdollisuudet **raportointiin** esitellään kunkin mallinnusvälineen kuvauksessa. Raportointia ei huomioida arvioinnissa, sillä tilastotietojen esittäminen ei Sovellusprojektien kannalta ole oleellinen piirre. Sillä voi ajatella olevan enemmän käyttöä mikroyrityksissä, mutta merkitys lienee joka tapauksessa mallinnusvälinettä valittaessa vähäinen.

Luvussa 5.2.3 esiteltyjä **koodin generointia ja takaisinmallinnusta** ei huomioida ohjelmistojen vertailussa, sillä Sovellusprojektien ja mikroyritysten projektien tarpeet vaihtelevat tuettavien ohjelmointikielten osalta. Lisäksi kattava vertailu lisäisi tutkielman työmäärää. Tuki koodin generoinnille ja takaisinmallinnukselle kuvataan kunkin mallinnusvälineen dokumentaation perusteella arvioinnin yhteydessä.

Sovellusprojekteissa laaditaan dokumentaatiota, johon on tarpeen liittää kuvatiedostoja kaavioista. **Kaavioiden tallentaminen grafiikan esittämiseen sopivissa tiedostomuodoissa** kokeillaan tutkielmassa kaikissa välineen tukemissa muodoissa. Tallenteen laatua arvioidaan vertaamalla kuvatiedostoa mallinnusvälineen työaseman ruudulla esittämään kaa-vioon.

Vertailussa huomioidaan mahdollisuus JPG-, BMP- ja PNG-tiedostomuodoissa olevien kuvien sekä Word- ja PDF-muotoisten asiakirjojen muodostamiseen mallin pohjalta. HTML-dokumentaation muodostamisen mahdollistavan toiminnon olemassaolo huomioidaan vertailussa. Paperitulostuksen mahdollisuus huomioidaan vertailussa esikatselun ja arkille sovitusmahdollisuuden olemassaolon osalta.

Luvussa 5.2.5 kuvattua **mallien ja kaavioiden siirtämistä** voisi mahdollisesti ajatella hyödynnettävän Sovellusprojekteissa ja mikroyrityksissä. Siirtämisen kattava kokeilu vaatisi kuitenkin todennäköisesti merkittävästi enemmän aikaa kuin tutkielmaan on käytettävissä, joten siirtämistä ei kokeilla. Välineen dokumentaatiossa esille tuodut siirtomahdolli-

suudet kuitenkin kerrotaan kunkin mallinnusvälineen kuvauksessa. Tuki XMI-tiedostomuodolle esitetään vertailtujen mallinnusvälineiden kuvausten yhteydessä.

5.3 Käytettävyyteen liittyvät vertailukohteet

Luvussa 4.2 kuvatun muodollisen käytettävyyden arvioinnin suorittaminen pienessä organisaatiossa on epärealistista, eikä tule kyseeseen myöskään pro gradu -tutkielman laajuuden huomioiden. Tutkielmassa onkin päädytty käyttämään menetelmänä käytettävyyden heuristista arviointia tutkielman tekijän toimiessa asiantuntijana.

Kuten luvussa 4.2 todettiin, asiantuntijan kokemuksella on vaikutusta löydettävien käytettävyysongelmien määrään. Tutkielman tekijällä on vähäinen kokemus käytettävyyden arvioinnista. Sovellusalueesta, eli UML-mallien laatimisesta ja mallinnusvälineistä, kokemusta on useiden vuosien ajalta. Löydettyjen käytettävyysongelmien määrän voi näin ollen arvioida sijoittuvan lähemmäs 22:a kuin 60 prosenttia kaikista käytettävyysongelmista.

Asiantuntija-arvioinnin rinnalla on käytetty liitteen 2 tarkistuslistaa, jonka perusteella tiettyjen toimintojen olemassaolo on tarkastettu johdonmukaisesti kaikista arvioiduista mallinnusvälineistä. Siihen on koottu opittavuuden, tehokkuuden, muistettavuuden ja virheettömyyden kannalta oleellisia toimintoja. Toiminnot on johdettu pääosin artikkelissa [Kirchner, s. 7–9 ja 14] esitettyjen periaatteiden mukaan.

Liitteessä 3 on esitetty heuristisen arvioinnin perusteella tehdyt havainnot arvioitujen mallinnusvälineiden käytettävyydestä. Arvioinnin aikana todettiin, että luvun 4.3 asteikosta puuttuu positiivisten havaintojen luokka. Koska ne kuitenkin vaikuttavat arviointiin, on ne huomioitu taulukossa.

Luvuissa 5.3.1–5.3.5 käsitellään niitä lähteissä [Auer], [Eichelberger], [Eriksson], [Fowler], [Khaled], [Kirchner] ja [Rumbaugh] esitettyjä vertailukohtia, jotka liittyvät käytettävyyteen. Luvussa 4.1 esitellyistä käytettävyyden ulottuvuuksista lähteissä nousivat esille opittavuus, tehokkuus ja virheettömyys. Muistettavuuteen ja tyytyväisyyteen liittyviä vertailukohtia lähteissä esitettiin vain muutamia.

5.3.1 Opittavuus

Uuden käyttäjän on opittava käyttämään järjestelmää riittävästi ennen kuin voi työskennellä sillä. Artikkelissa [Auer] pohditaan helppoa opittavuutta ja käyttöönottoa erityisesti ajatellen mallinnusvälineiden käyttöä **oppilaitoksissa**. Pienen UML-mallin luonnostelua varten käyttöliittymän tulisi olla intuitiivinen, eikä käyttäjää pitäisi häiritä harvoin tarvittavilla ominaisuuksilla. Artikkelissa todetaan myös, että monet työkalut ovat ominaisuuksiltaan monipuolisia ja tukevat laajalti UML:a, mutta epäonnistuvat UML:n opetuksen, ohjelmistokehitysprosessin ja ketterien menetelmien tukemisessa.

Työkaluohjelmistojen tekijät pyrkivät artikkelin [Auer, s. 3] mukaan tukemaan mahdollisimman suurta määrää **UML-kielen ominaisuuksia**, minkä nähdään vaikeuttavan ohjelmistojen käytön oppimista. Toisaalta vuodelta 2009 olevan artikkelin [Eichelberger, s. 49] mukaan markkinoilla olevista työkaluohjelmista vain muutama on suurelta osin yhteensopiva UML-standardin kanssa. Käytön vaikeus johtuneekin siten muista syistä kuin varsinaisista UML-kielen vaatimuksista.

Artikkelissa [Kirchner, s. 7] oppimista helpottavana seikkana nähdään ohjelman **käyttöliittymän yhdenmukaisuus** käyttöjärjestelmän tavanomaisten toimintojen kanssa. Esimerkiksi, jos sellaiset yleiset toiminnot kuin kopiointi, poisto ja liittäminen on toteutettu Windows-ympäristössä totutulla tavalla, kokenut Windows-käyttäjä voi keskittyä niiden sijasta ohjelmistolle erityisten toimintojen opetteluun. Tässä suhteessa ongelmallisia ovat esimerkiksi ohjelmistot, jotka on alun perin suunniteltu eri käyttöjärjestelmäympäristöön kuin niitä käytetään.

Mallinnusvälineen tulisi artikkelin [Kirchner, s. 13] mukaan tarjota käyttäjälle **ohjelmadokumentaation** lisäksi erilaisia hyvin **dokumentoituja esimerkkejä**, koska niiden avulla ohjelman käyttömahdollisuuksien arvioiminen ja käytön oppiminen on helpompaa. Lisäksi olisi toivottavaa, että esimerkkien mukaisten mallien laatimisesta olisi laadittu vaiheittain etenevä opastus. Esimerkkien olemassaolo huomioidaan vertailussa.

Myös mallinnusvälineen **piirto-ominaisuuksien** voi ajatella osaltaan helpottavan tai vaikeuttavan välineen käytön opettelua. Piirto-ominaisuuksia on käsitelty tarkemmin luvussa 5.3.2.

5.3.2 Tehokkuus

Kuten luvussa 4.1 todettiin, opittuaan käyttämään järjestelmää kokeneen käyttäjän tulisi kyetä suoriutumaan tehtävistään tuloksekkaasti ja nopeasti. Mallinnusvälineen ohjainten sijoittelun tulisi olla johdonmukaista ja kunkin toiminnon suorittamiseen vaadittavien **työvaiheiden määrän** tulisi olla suhteessa niiden käytön tiheyteen. Artikkelissa [Auer, s. 3] on kiinnitetty huomiota siihen, että usein monia työvaiheita vaatii jopa perustoimintojen suorittaminen, kuten elementin ominaisuuksien lisääminen kaavioon.

Kirjassa [Eriksson, s. 32] todetaan, että elementtien tehokkaiden valinta-, sijoittelu-, yhdistely- ja määrittelyominaisuuksien lisäksi **ohjelmiston tulisi tukea käyttäjää sääntöjen mukaisen kaavion piirtämisessä**. Välineen olisi tuettava elementtien sijoittelua kaavioissa siten, että viestiviivat eivät kulje toistensa yli. Tämä voidaan toteuttaa joko automaattisesti tai antamalla käyttäjän itsensä sijoitella elementit tarkoituksen mukaisella tavalla.

Kirjassa [Rumbaugh, s. 109] huomautetaan, että **monimerkityksisen tai määrittelemättömän informaation** salliminen on tarpeen erityisesti mallinnuksen alkuvaiheessa. Vaikka tavoitteena olisi kattava ja johdonmukainen malli, sen tekeminen ei yleensä ole kerralla mahdollista. Lisäksi varsinkin mallinnuksen alkuvaiheessa jonkin elementin yksittäisellä ominaisuudella ei välttämättä ole suunnittelun kannalta merkitystä, joten elementtejä pitäisi voida lisätä ilman, että kaikille ominaisuuksille on määriteltävä arvo. Muutoinkin työkalun tulisi sallia mallin tarkkuuden vaihtelu.

Artikkelissa [Auer, s. 3–5] kiinnitetään erityistä huomiota muutamiin käyttöliittymän tehokasta käyttöä koskeviin seikkoihin:

- Modaalisten ponnahdusikkunoiden avulla toteutettu elementtien muokkaaminen on usein hidasta ja hankalaa.
- Mallinnusvälineiden työkalupalkeissa käytetyt kuvakkeet ovat usein pieniä, eivätkä niissä käytetyt kuvat muistuta juurikaan vastaavia UML-elementtejä.

- Lisäksi nähdään hyvänä myös kirjassa [Rumbaugh, s. 109] mainittu mahdollisuus antaa elementille lisäämisen yhteydessä vain tarvittavat attribuutit.

Malli koostuu tyypillisesti useista toisiinsa linkittyvistä kaavioista. Kirjassa [Eriksson, s. 32] ja artikkelissa [Kirchner, s. 14] todetaan, että **siirtyminen kaavioiden, elementtien ja esitystasojen välillä** tulisi olla mahdollisimman helppoa ja selkeää siten, että käyttäjä on jatkuvasti tietoinen siitä, mitä osaa mallista tarkastelee. Samoin **kaavioiden selailun ja elementtien hakemisen** tulisi olla työkalussa mahdollista.

Navigointia helpottaviksi keinoiksi mainitaan elementteihin liitettävät hyperlinkit, esitettävän informaation suodattaminen ja piilottaminen sekä näkymien määrittäminen. Artikkelissa [Kirchner, s. 14] esitetään, että hyvin toimiva automaattinen asettelutoiminto ja suurennusmahdollisuus mahdollistavat kaavioiden tarkastelemisen halutulla tarkkuudella. Kirjassa [Rumbaugh, s. 109] nostetaan esille myös mahdollisuus erilaisten visualisointien käyttämiseen näyttötekniikan salliessa.

Artikkelissa [Kirchner, s. 8] laatua parantavaksi tekijäksi mainitun **dokumentaation** voi katsoa tukevan ohjelmiston opittavuutta, mutta toisaalta hyvä tilannekohtainen ohje lisää myös käytön tehokkuutta. Opastustoiminnot ja dokumentaatio ovat myös yhtenä kohtana luvussa 4.2 esitellyssä Nielsenin listassa.

Mallinnusvälineestä tulisi olla tarjolla **dokumentaatiota eri käyttäjäryhmille**, kuten ylläpitäjälle, mallintajalle ja toteuttajalle. Dokumentaatiota tulisi täydentää käytönaikaisella toimintokohtaisella ohjeella, jossa vähintään viitataan oikeaan kohtaan dokumentaatiossa.

5.3.3 Muistettavuus

Muistettavuudella on erityistä merkitystä järjestelmän satunnaisille käyttäjille. Ainakin luvussa 5.3.1 mainittu ohjelman käyttöliittymän **yhdenmukaisuus** käyttöjärjestelmän tavanomaisten toimintojen kanssa ja luvussa 5.3.2 mainittu **ohjainten johdonmukainen sijoittelu** edesauttavat myös muistettavuutta.

Muistettavuuteen liittyy myös artikkelissa [Kirchner, s. 7] esitetty toivomus, että käyttäjä voisi jossain määrin **muokata käyttöliittymää** omiin työtapoihinsa sopivaksi. Näin käyttäjä voi sijoitella usein käyttämänsä toiminnot helposti saataville.

5.3.4 Virheettömyys

Artikkeleissa [Khaled, s. 112] ja [Kirchner, s. 9] todetaan, että **ohjelmiston vakaus** on merkittävä tekijä ohjelmiston käytettävyyttä arvioitaessa. Riippumatta mahdollisen epävakauden syystä ohjelmiston tulisi kyetä **estämään tietojen katoaminen**, mikäli se kaatuu kesken työskentelyn. Mallinnusvälineen tulisi artikkelin [Kirchner, s. 7] mukaan kyetä käsittelemään **käyttäjän virheellisiä syötteitä** ilman, että se joutuu määrittelemättömään tilaan tai kaatuu.

5.3.5 Tyytyväisyys

Järjestelmän käyttämisen miellyttävyys riippuu käyttäjän subjektiivisesta kokemuksesta. Erikssonin [Eriksson, s. 32] toteamus, että **kaavioiden piirtämisen tulisi olla helppoa ja hauskaa**, on suorin tyytyväisyyteen liittyvä viittaus, joka tutkielman lähteissä mainittiin. Myös luvuissa 5.3.1 ja 5.3.2 kuvatun artikkelissa [Auer] esitetyn mallinnusvälineiden opittavuuden ja tehokkuuden kritiikin voi katsoa liittyvän tyytyväisyyteen.

Käyttäjätuen mainitaan artikkelissa [Kirchner, s. 8] vaikuttavan ohjelmiston laatuun. Käyttäjätuki vastannee avoimen lähdekoodin yhteydessä lähinnä yhteisön keskustelufoorumeilla antamaa tukea, kuten artikkelissa [Lakhani, s. 924] esitetään.

5.3.6 Huomioitavat käytettävyyteen liittyvät vertailukohteet

Käytettävyyden arviointi perustuu ensisijaisesti luvussa 4.2 esitettyyn Nielsenin listaan. Sen tueksi on laadittu ominaisuuksista liitteen 2 tarkistuslista, joka pohjautuu lähteistä [Auer], [Khaled], [Eichelberger], [Eriksson], [Kirchner], [Koskimies] ja [Rumbaugh] poimittuihin käytettävyyteen liittyviin piirteisiin. Tarkistuslistan kohdat on valittu siten, että ne ovat tutkielman tekijän yksin todettavissa.

Luvussa 4.1 kuvattuun tehokkuuteen liittyen vertailukohdaksi oli artikkelin [Auer] pohjalta tarjolla elementtien lisäämiseen tarvittavien työvaiheiden määrä. Nopeasti osoittautui, että kaikissa vertailuun valituissa ohjelmistoissa elementti on valittavissa suoraan työkalupal-

kista, joten se poistettiin vertailukohtien joukosta. Arvioinnissa on kuitenkin otettu kantaa elementtien lisäämisen sujuvuuteen kunkin ohjelmiston osalta.

Dokumentaation osalta huomioidaan vertailussa toimintokohtaisen ohjeen, sovellusohjeen, käsikirjan ja usein kysyttyjen kysymysten listan olemassaolo. Mallinnusvälineiden vertailussa todetaan kunkin välineen sourceforge.net-sivustolla ilmoittaman tukikanavan kautta lähetettyihin viesteihin saatujen ratkaisujen määrää edellisen kuuden kuukauden aikana.

Mallinnusvälineen virheettömyys on sen käytettävyyden kannalta merkittävä tekijä. Vakauden arvioiminen luotettavalla koejärjestelyllä vaatisi merkittävästi enemmän aikaa kuin siihen tutkielmassa on käytettävissä. Mahdolliset käytön aikana havaitut ja ohjelmistojen dokumentaatiosta löydetyt merkittävät ongelmat tuodaan esiin ohjelmistojen vertailussa.

Tyytyväisyyden tarkastelu perustuu subjektiiviseen arvioon, sekä sen tulos voi vaihdella arvioijan tavoitteista ja taustasta riippuen. Tutkielmassa kunkin arvioidun mallinnusvälineen osalta tyytyväisyydestä on arvioinnin yhteydessä esitetty tekijän arvioinnin aikana syntynyt subjektiivinen näkemys

5.4 Mallinnusvälineen elinkaarikustannukset

Elinkaarikustannukset olivat vertailukohde, jonka huomioimista tutkielmassa toivottiin Sovellusprojekteihin liittyen. Vaikka avoimen lähdekoodin sovellusten hankinnasta ei aiheudu merkittäviä kuluja, voivat esimerkiksi erot ylläpidon työmäärissä aiheuttaa merkittäviä eroja elinkaarikustannuksissa. Kustannuksia ei tutkielmassa ole arvioitu rahamäärinä, vaan on pyritty arvioimaan kunkin mallinnusvälineen ylläpidon vaatimaa työmäärää ja mahdollisesti laitteistovaatimuksia.

5.4.1 Suorat ja välilliset kustannukset

Artikkelissa [Kirchner, s. 6] käsitellään mallinnusvälineen elinkaarikustannuksia yleisellä tasolla. **Suoriksi elinkaarikustannuksiksi** todetaan hankintahinnan lisäksi laitteistovaatimuksista seuraavat kustannukset.

Asennuksen ja jakelun helppous nostetaan esille artikkelissa [Auer, s. 2–3]. Nopeasti ja helposti asennettavissa oleva sovellus on mahdollista asentaa uudelleen tai uuteen laittee-

seen aina tarvittaessa. Asennusta ja jakelua vaikeuttaviksi seikoiksi mainitaan asennuspaketin puuttuminen ja suuri tiedostokoko, lisenssiavaimen jakelu, riippuvuus tietystä käyttöjärjestelmästä sekä ohjelmiston suuret vaatimukset laitteiston tehokkuudelle. Tuki useille käyttöjärjestelmille todetaan artikkelissa [Kirchner, s. 9] mahdolliseksi arviointikohteeksi.

Asennuksen yhteydessä käyttäjän tulisi voida artikkelin [Kirchner, s. 8] mukaan asettaa ohjelmaan tarvettaan vastaavat määritykset, kuten käytettävä kieliversio ja asennettavat komponentit. Myös asennuksen poistamista varten tulisi olla tarjolla toiminto.

Lisäksi **mallinnusvälineen välillisiin elinkaarikustannuksiin** sisältyvät ainakin luvussa 5.3.1 esitelty opittavuus ja luvussa 5.3.5 esitelty käyttäjätuki. Näihin liittyvät kustannukset muodostuvat ohjelmiston käytön opettelun ja ongelmatilanteiden ratkaisemiseen käytetystä työajasta.

5.4.2 Huomioitavat elinkaarikustannusten vertailukohteet

Avoimen lähdekoodin ohjelmistojen **hankintakustannus** ei yleensä ole merkittävä ja vertailukohtana käytetty suljetun lähdekoodin ohjelmisto valittiin siten, että siitä on saatavilla maksuton kokeiluversio. Mallinnusvälineiden elinkaarikustannuksia arvioitaessa täytyykin tarkastella muita seikkoja kuin hankintahintaa.

Sovellusprojektit opintojakson aikana työryhmän jäsenillä voi olla tarve asentaa väline useille eri laitteille, joten **asennuksen helppouden** tulee olla tarkasteltavana vertailukohtana. Arvioinnissa esiteltäviä seikkoja ovat asennuspaketin koko ja asennusprosessi, sekä käyttöjärjestelmätuki. Lisäksi **laitteistovaatimukset** huomioidaan, jos ne ovat jollain tavoin muista vertailluista välineistä poikkeavat. Ohjelmistojen elinkaarikustannuksiin vaikuttavia ominaisuuksia on esitetty liitteessä 6.

6 Mallinnusvälineiden arviointi

Tutkielman tavoitteena on arvioida avoimen lähdekoodin UML-mallinnusvälineiden soveltuvuutta opiskelijaprojektien käyttöön. Vertailussa on mukana myös yksi suljetun lähdekoodin sovellus, josta on saatavilla maksutta rajoittamaton kokeiluversio. Sen avulla on tarkoitus selvittää, ovatko suljetun lähdekoodin ohjelmistot selkeästi avoimen lähdekoodin sovelluksia kehittyneempiä tutkielmassa tarkasteltavien vertailukohtien osalta.

Luvussa 6.1 kuvataan ohjelmistojen valintaa, sekä esitellään lyhyesti vertailun ulkopuolelle rajattuja ohjelmistoja ja kerrotaan perusteet, jolla rajaus on kunkin osalta tehty. Luvuissa 6.2.1–6.2.3 esitellään vertailtavat avoimen lähdekoodin ohjelmistot ja luvussa 6.2.4 vertailukohtana käytetty suljetun lähdekoodin ohjelmisto. Luvussa 6.3 kuvataan, kuinka vertailu suoritettiin. Vertailua tehtäessä mallinnettu esimerkkijärjestelmä on esitelty luvussa 6.4. Luku 6.5 käsittelee vertailun tuloksia, ja luvussa 6.6 esitetään tulosten pohjalta tehdyt johtopäätökset ja suositukset.

6.1 Ohjelmistojen kartoitus, rajaus ja valinta

Arvioitavia ohjelmistoja kartoitettiin sourceforge.net-sivustolta, joka on laaja avoimen lähdekoodin ohjelmistoprojekteja esittelevä sivusto. Tavoitteena oli löytää vertailtavaksi 3–5 ohjelmistoa, jotka kuvauksensa perusteella näyttäisivät sopivan Sovellusprojekti-opintojaksolla käytettäviksi.

6.1.1 Kartoituksen ja rajauksen suorittaminen

Ohjelmistoja haettiin sourceforge.net-sivuston omalla hakutoiminnolla. **Hakutulosta rajatiin** käyttämällä `Freshness`-suodattimen arvoa `Resently updated`, joka rajaa haun viimeisen kahden vuoden aikana päivitettyihin projekteihin. Suotimen käytöllä haluttiin varmistaa, että hakuosumien joukkoon ei nouse ohjelmistoja, joiden kehitys on selkeästi pysähtynyt.

Hakusanalla `UML` saatiin tulokseksi 105 osumaa. Hakusanalla `UML tool` löytyi 47 ohjelmistoa, ja sanalla `UML modeling tool` ohjelmistoja löytyi 28. Tarkemmat hakusanat kuitenkin näyttivät rajaavan hakutuloksesta pois myös mahdollisesti vertailuun soveltuvia

ohjelmistoja, sillä esimerkiksi Violet UML Editor -nimisen ohjelmiston kuvauksessa ei esiinny sanoja *tool* ja *modeling*.

Sourceforge.net-sivuston hakutoiminto mahdollistaa **hakutuloksen rajaamisen** useilla suotimilla. Suotimien ongelmana näytti hakuosumien määrän perusteella olevan se, että suodatuksen kohteena olevaa metatietoa ei ollut määriteltynä kaikille ohjelmistoille, minkä vuoksi ne rajautuivat pois hakutuloksesta. Tästä syystä haku tehtiin vain käyttäen hakusanaa `UML`, sekä suodatinta `Freshness` arvolla `Resently updated`. Työmäärän vähentämiseksi viimevaiheessa käytettiin lisäksi suodinta `Category` arvolla `Software Development`, joka ei nopean tarkastelun perusteella näyttänyt karsivan hakutuloksista vertailuun soveltuvia ohjelmistoja. Tällöin hakuosumia oli 68.

Hakutuloksesta karsittiin seuraavaksi ohjelmistot, jotka lyhyen kuvauksensa perusteella olivat selkeästi todettavissa ajateltuun käyttötarkoitukseen kelpaamattomiksi. Karsitut ohjelmistot joko tukivat vain yksittäisiä kaaviotyyppejä tai eivät olleet UML-mallinnustyökaluja, vaan esimerkiksi koodigeneraattoreita.

Jäljelle jääneet ohjelmistot olivat

- ArgoUML,
- DoUML,
- Gaphor,
- Modelio - Modeling environment (UML),
- PlantUML QEditor,
- Taylor,
- Umbrello UML Modeler,
- UML-Editor,
- WhiteStarUML ja
- Violet UML Editor.

Näistä 10 ohjelmistosta valittiin arvioitaviksi ne, jotka **kuvauksensa perusteella** arvioituna todennäköisesti vastaisivat parhaiten Sovellusprojektien tarpeita. Valintaan vaikuttivat oleellisesti ohjelmiston kehitysvaihe ja tuettujen kaaviotyyppien määrä.

Huomion arvoista on, että kuvausten perusteella ei aina ole mahdollista suoraan saada selville, mitä UML-standardin versiota ohjelmisto tukee. Joissain tapauksissa tuetun version voi tosin päätellä tuettujen kaaviotyyppien nimistä vertaamalla niitä luvun 3.1.2 taulukossa 1 esitettyihin nimiin.

Arvioitaviksi valitut ohjelmistot ArgoUML, Modelio - Modeling environment, ja WhiteStarUML on esitelty luvuissa 6.2.1–6.2.3. Luvussa 6.2.4 esitellään vertailukohtana käytetty suljetun lähdekoodin ohjelmisto IBM Rhapsody Modeller.

Valitsematta jääneistä ohjelmistoista selkeimmin pois rajautuivat toiminnallisuuksiensa rajallisuuden perusteella PlantUML Qeditor, UML-Editor ja Taylor. DoUML, Gaphor, Umbrello UML Modeller ja Violet UML Editor olisivat voineet kuvaustensa perusteella olla mahdollisia vertailtavia ohjelmistoja. Erityisesti Gaphor ja Violet UML Editor vaikuttivat lähes samanveroisilta vertailuun valittujen ohjelmistojen kanssa. Ne jätettiin kuitenkin valitsematta Windows-versioidensa asennukseen ja vakauteen liittyvien puutteiden vuoksi.

6.1.2 Vertailun ulkopuolelle rajatut ohjelmistot

DoUML on kuvauksensa [de Freitas] mukaan johdettu BoUML-projektista. Sen kerrotaan olevan UML 2 -mallinnusväline, joka kykenee koodin generointiin, takaisinmallinnukseen ja edestakaisinmallinnukseen. Se tukee ainakin luokka- ja tilakaavioita. Koodin generoiminen on mahdollista ohjelmointikielillä C++, Java, Python, PHP ja IDL. Mallin tallentaminen XMI-muodossa on mahdollista. Kuvauksensa perusteella DoUML olisi ollut ominaisuuksiltaan mahdollinen vertailtava mallinnusväline. Sitä ei kuitenkaan valittu, koska ohjelmisto on alpha-vaiheessa ja käyttäjille suunnattu dokumentaatio oli hyvin vähäistä.

Gaphor on ominaisuuskuvauksensa [Gaphor] mukaan mallinnusympäristö, jonka tietomalli on yhteensopiva UML:n version 2.0 kanssa. Se tukee luokka-, komponentti-, käyttötapaus-, toiminto-, vuorovaikutus-, tila- ja profiilikaavioita. Gaphoria voidaan käyttää sekä Unix- että Windows-ympäristöissä. Sen osalta rajaaminen pois arvioinnista perustui lähinnä valittuja ohjelmistoja monimutkaisempaan Windows-asennukseen, mikä olisi vaikuttanut ohjelmiston elinkaarikustannuksiin.

PlantUML Qeditor on kuvauksensa [Borcoman] perusteella yksinkertainen editori käytettäväksi PlantUML-ohjelmiston kanssa. Sitä ei valittu vertailuun, koska se muodostaa kaavioiden graafiset esitykset tekstimuotoisen kuvauksen perusteella, eli kaavioiden piirtäminen suoraan piirtotyökaluilla ei ole mahdollista. Se ei myöskään ole itsenäinen ohjelmisto.

Taylor on kuvauksensa [Gilbert] perusteella suunniteltu tukemaan yritysjärjestelmien kehittämistä malliperustaista arkkitehtuuria hyödyntäen. Taylor koostuu Eclipse-kehitysympäristön liitännäisistä. Sitä ei valittu vertailuun, koska sen toimintoja on erikoistumisen vuoksi rajoitettu suhteessa yleiskäyttöisempiin mallinnusvälineisiin. Se ei myöskään ole itsenäinen ohjelmisto.

Umbrello UML Modeller on projektin kotisivun [Umbrello] mukaan KDE-yhteisön ohjelmistokokonaisuuteen liittyvä UML-kaavioiden laatimiseen tarkoitettu ohjelmisto. Se tukee UML-kaavioista käyttötapaus-, luokka-, komponentti-, sekvenssi-, vuorovaikutus-, tila-, aktiviteetti- ja sijoittelukaavioita. Koodin generoiminen on mahdollista useille ohjelmointikielille, kuten Java, PHP, C++ ja C#. Ohjelmisto jätettiin valitsematta vertailuun, koska siitä ei ollut dokumentaation perusteella saatavilla arvioinnin tekohetkellä vakaasti toimivaa Windows-versiota.

UML-Editor-projektin tavoitteena on kehittää mallinnusväline, jonka helppokäyttöisyys edesauttaa malliperustaisen ohjelmistoprosessin leviämistä. Ohjelmisto jätettiin valitsematta arvioitavaksi, koska dokumentaation [Hirzel, s. 7] perusteella sen tuki kaaviotyypeille näyttää rajoittuvan luokkakaavioon ja mahdollisesti pakkauskaavioon.

Violet UML Editor on esittelynsä [Horstmann] mukaan helppokäyttöinen, useissa käyttöjärjestelmissä toimiva UML-editori. Se tukee käyttötapaus-, luokka-, aktiviteetti-, sekvenssi-, tila- ja oliokaavioita. Ohjelmisto painottaa helppokäyttöisyyttä, eikä pyri UML-standardin kaikkien piirteiden noudattamiseen. Se jätettiin valitsematta vertailuun, koska tuettujen kaaviotyyppien määrä on vähäisempi kuin vertailuun valituissa ohjelmistoissa. Lisäksi osa ominaisuuksista on käytettävissä vain, jos ohjelmisto liitetään Eclipse-kehitysympäristöön.

6.2 Arvioidut ohjelmistot

Luvuissa 6.2.1–6.2.3 esitellään arvioidut avoimen lähdekoodin ohjelmistot. Luvussa 6.2.4 on esitelty vertailukohtana käytetty suljetun lähdekoodin ohjelmisto.

6.2.1 ArgoUML

ArgoUML on kuvauksensa [Tigris.org] mukaan johtava avoimen lähdekoodin UML-mallinnusväline. Se tukee kaikkia UML-version 1.4 kaaviotyyppejä. Ohjelmistosta on saatavilla kymmenen kieliversiota, jotka ovat amerikan englanti, brittienglanti, ranska, saksa, italia, portugali, espanja, venäjä, norja ja kiina.

Java-alustalla toimivaa ArgoUML:ää voidaan käyttää sellaisissa käyttöjärjestelmäympäristöissä, joihin Java-virtuaalikone on saatavilla. Työasemaan asennettavan version lisäksi se on ajettavissa verkossa. Arvioinnissa on käytetty työasemaan asennettavaa versioita.

ArgoUML:llä on mahdollista tallentaa kaavioita tiedostomuodoissa PNG, GIF, SVG, PS ja EPS. Kokeiltaessa kaavion tallennusta eri tiedostomuodoissa havaittiin, että EPS-tiedostoa katseltaessa kaikki assosiaatioviivat eivät näkyneet ja GIF-tiedostomuodossa osa viivoista oli epätarkkoja.

Tulostuksen esikatselua ei ArgoUML:ssa ole. Arkin suuntaa on mahdollista kääntää ja marginaalien leveyttä voidaan säätää.

Dokumentaation generoiminen ei ole mahdollista, eikä tilastotietoa mallista ole saatavilla. Ohjelmistolla laaditut mallit tallennetaan XMI-tiedostomuodossa. Tallennuksessa käytettävä XMI-versio on 1.2. Mallien tuominen on mahdollista sekä XMI 1.1 että XMI 1.2 -muodoissa.

Koodin generointi on mahdollista rakennekaavioista ohjelmointikielille Java, C++, C#, PHP4 ja PHP5. Takaisinmallinnusta tuetaan oletuksena Java-lähdekoodista luokkakaavion osalta.

6.2.2 Modelio - Modeling environment

Modelio on kuvauksensa [Modelio Community 2013a] perusteella avoimen lähdekoodin mallinnusympäristö, joka on joustavasti laajennettavissa erillisillä komponenteilla. Se tukee UML-standardin versiota 2. Liiketoimintaprosessien kuvaaminen on mahdollista BPMN-kaavioin (Business Process Model and Notation). Erillisinä moduuleina on saatavilla tuki ainakin SysML-, TOGAF- ja SoaML-mallinnustekniikoille.

Ohjelmistosta on ladattavissa asennuspaketti Windows- ja Linux-käyttöjärjestelmille. Saatavilla ovat sekä 32-bittiset että 64-bittiset versiot.

Modeliosta on mahdollista tallentaa kaavioita tiedostomuodoissa BMP, GIF, JPG ja PNG. Lisäksi on mahdollista kopioida kaavio kuvana käyttöjärjestelmän leikepöydälle, josta se voidaan liittää esimerkiksi kuvankäsittelyohjelmaan. Kokeiltaessa kaavioiden tallentamista GIF-tiedoston tallennus ei toiminut.

Tulostustoiminnon yhteydessä on mahdollisuus tulostuksen esikatseluun. Kaavio voidaan sovittaa paperiarkille, ja arkin suuntaa on mahdollista kääntää.

Dokumentaation generoiminen ei ole mahdollista, eikä tilastotietoa mallista ole saatavilla. Modelio tukee mallien viemistä ja tuomista XMI 2.4 -muodossa. Koodin generoiminen on mahdollista Java-ohjelmointikielelle. Takaisinmallintaminen on mahdollista tehdä Java-lähdekoodista. Erillisinä moduuleina on saatavilla laajennoksia eri kielille, kuten C++ ja C#.

6.2.3 WhiteStarUML

WhiteStarUML on beta-kehitysvaiheessa oleva StarUML-nimiseen avoimen lähdekoodin mallinnusvälineeseen pohjautuva ohjelmisto. Sen kotisivulla [Szpilewski 2013b] esillä oleva informaatio on varsin vähäistä, mutta sivuilta on ladattavissa tarkemman kuvauksen sisältävä dokumentti [Szpilewski 2013a].

Ohjelmisto tukee UML-versiota 1.4, mutta dokumentaatio toteaa sen hyväksyvän myös UML-version 2.0 mukaisen notaation käyttämisen. Toimiakseen ohjelmisto vaatii vähin-

tään Windows XP -käyttöjärjestelmän, johon on asennettu .NET 2.0. Suositeltuja käyttöjärjestelmiä ovat Windows 7 ja 8.

WhiteStarUML:sta on mahdollista tallentaa kaavioita tiedostomuodoissa BMP, JPG, EMF ja WMF. Tulostustoiminnon yhteydessä on mahdollisuus tulostuksen esikatseluun. Kaavio voidaan sovittaa paperiarkille ja yhdelle arkille voidaan tulostaa useita kaavioita. Arkin suuntaa on mahdollista kääntää ja marginaalien leveyttä voidaan säätää. Ylä- tai alatunnisteeseen voidaan tulostaa muutamia perustietoja, kuten kaavion ja projektin nimi.

Dokumentaation generointi valmiita raporttipohjia käyttäen on mahdollista, mutta tilastotietoa mallista ei ole saatavilla. Koodin generoiminen on mahdollista erillisiä moduuleita käyttäen. Mallien siirtäminen on mahdollista käyttäen XMI-muotoa.

6.2.4 Rational Rhapsody Modeler

IBM Rational Rhapsody Modeler valikoitui arvioitavaksi suljetun lähdekoodin ohjelmistoksi useammasta syystä. Ensinnäkin sen on kehittänyt luvussa 3.1.1 mainittu, ja nykyisin IBM:n omistama, vahvasti UML:n muodostamisen taustalla ollut Rational Software. Toiseksi ohjelmistosta oli saatavilla myös maksuton kokeiluversio, jonka toimivuutta ei kuvauksen [IBM 2013a] perusteella ollut rajoitettu arvioitavien ominaisuuksien osalta. Kokeiluversion rajoitukset koskevat integraatiota muihin Rational Rhapsody -tuoteperheen ohjelmistoihin. Käytännössä esimerkiksi ohjelmakoodin generoiminen edellyttää integraatiota.

Modeler tukee UML:n versiota 2.1. Ohjelmisto vaatii toimiakseen Windows-käyttöjärjestelmän. Siitä on saatavilla kieliversiot englanniksi, saksaksi, japaniksi, ranskaksi, ruotsiksi ja espanjaksi.

Ohjelmistosta on mahdollista tallentaa kaavioita tiedostomuodoissa BMP, EMF, JPG ja TIFF. Raportointitoiminto muodostaa kirjallisen yhteenvedon mallista, mutta raportti ei sisällä tilastotietoa.

Tulostettaessa kaavio voidaan sovittaa paperiarkille, arkin suuntaa on mahdollista kääntää ja marginaalien leveyttä voidaan säätää. Tulostuksen esikatselumahdollisuus on käytettä-

vissä. Ylä- tai alatunnisteeseen voidaan tulostaa muutamia perustietoja, kuten kaavion ja projektin nimi.

Mallien siirtäminen on mahdollista käyttäen XMI-versioita 1.0, 1.1, 1.2 ja 2.1. Käyttöliittymässä on valittavissa toiminto C++-ohjelmakoodin generoimiseksi, mutta se ei testiympäristössä toiminut.

Ohjelmistosta oli yllättävän hankalaa saada taustatietoja ottaen huomioon, että se kuuluu kaupalliseen tuoteperheeseen. Tietojen saamista vaikeutti se, että esittelysivulla ollut linkki tuotetietoihin ei toiminut, eikä sivuston hakutoimintokaan niitä löytänyt. Tuen saamiseksi ohjattiin käyttäjäyhteisön keskustelupalstalle.

6.3 Vertailukohteiden soveltaminen vertailtaviin mallinnusvälineisiin

Avoimen lähdekoodin mallinnusvälineiden ominaisuuksien arviointi suoritettiin siten, että mallinnusvälineen Windows-versio ladattiin sourceforge.net-sivustolla ilmoitetusta lähteestä ja asennettiin Windows 7 -työasemalle. Seuraavaksi mallinnusvälineen kuvaustavat ja ominaisuudet käytiin läpi liitteiden 1 ja 2 tarkistuslistojen perusteella. Tämän jälkeen kutakin mallinnusvälinettä käyttäen laadittiin luvussa 6.4 esiteltyä esimerkkisovellusta kuvaava malli noudattaen mahdollisuuksien mukaan ennalta määriteltyä työkulkua.

Dokumentaation perusteella tarkasteltavien vertailukohteiden osalta tukeuduttiin sourceforge.net-sivustolla esitettyihin lähteisiin, sekä mahdollisesti ohjelmistoon liitettyihin ohjeisiin. Tehdyt havainnot on esitetty arvioinnin yhteydessä luvussa 6.5.3–6.5.5.

Suljetun lähdekoodin sovelluksen arvioinnissa meneteltiin vastaavalla tavalla kuin avoimen lähdekoodin välineitä arvioitaessa. Niistä poiketen ohjelmisto ladattiin ja dokumentaatiota haettiin valmistajan omalta sivustolta. Tehdyt havainnot on esitetty luvussa 6.5.6.

6.4 Mallinnusvälineiden arvioinnissa mallinnettu esimerkkijärjestelmä

Luvussa 4.2 esitelty käytettävyyden heuristinen arviointi on mahdollista suorittaa käyttämättä järjestelmää esimerkiksi kuvaruutukuviin tukeutuen, mutta arvioinnilla saatavat tulokset jäävät tällöin vajaiksi. Tutkielmassa tarkasteltavista mallinnusvälineistä on kuitenkin

käytettävissä toimivat ohjelmistot, joten niitä tarkasteltiin käytännössä laatimalla malli esimerkkijärjestelmästä.

Mallinnusvälineitä vertailtaessa kullakin niistä on laadittu kirjassa [Eriksson, s. 277–302] esiteltyä **kirjastotietojärjestelmää kuvaava malli**. Tutkielman malli ei täysin vastaa kirjassa esitettyä mallia, sillä tutkielmassa on pyritty käyttämään UML 2 -version kuvaustapoja, mikäli mallinnusväline on sen mahdollistanut. Lisäksi työmäärän pienentämiseksi samoja kaaviotyyppejä edustavista kaavioista on laadittu vain yksi olettaen niiden olevan tutkielmassa tarkasteltavien vertailukohteiden osalta toisiaan vastaavat.

Kirjastojärjestelmälle on kirjassa [Eriksson, s. 278] esitetty seuraavat vaatimukset:

- Järjestelmä on kirjaston tukijärjestelmä.
- Kirjasto lainaa kirjoja ja lehtiä lainaajille, jotka on rekisteröity järjestelmään, kuten kirjat ja lehdetkin.
- Kirjasto hoitaa uusien nimikkeiden ostamisen kirjastoon. Suosittuja nimikkeitä ostetaan useita kappaleita. Vanhoja kirjoja ja lehtiä poistetaan, kun ne vanhenevat tai menevät huonoon kuntoon.
- Kirjastonhoitaja on kirjaston työntekijä, joka on vuorovaikutuksessa lainaajien kanssa ja jonka työtä järjestelmä tukee.
- Lainaaja voi varata kirjan tai lehden, joka ei sillä hetkellä ole saatavilla kirjastossa. Hänelle kerrotaan, kun varattu nimike palautetaan tai ostetaan kirjastoon. Varaus perutaan, kun lainaaja lainaa kirjan tai lehden, tai jos varaus erikseen perutaan.
- Kirjasto voi luoda, päivittää ja poistaa tietoja nimikkeistä, lainaajista, lainoista ja varauksista.
- Järjestelmä toimii kaikilla suosituilla laitealustoilla, ja siinä on nykyaikainen graafinen käyttöliittymä.
- Järjestelmään voidaan helposti lisätä uusia ominaisuuksia.

Mallinnuksen työvaiheet toistettiin kullakin mallinnusvälineellä mahdollisimman yhdenmukaisesti. Ensin jokaista ohjelmistoa käyttäen laadittiin käyttötapauskaavio. Tämän jälkeen laadittiin luokka-, sekvenssi-, komponentti- ja tilakaaviot. Lopuksi laadittiin käyt-

töönottokaavio. Mallin kaaviot on esitetty kuvissa 4, 6, 10, 14, 15 ja 16 luvuissa 3.3.1–3.3.7.

6.5 Mallinnusvälineiden vertailu

Luvussa 6.2 esiteltyjen mallinnusvälineiden soveltuvuutta käytettäväksi Sovellusprojekti-opintojaksolla ja mikroyrityksissä vertailtiin luvussa 5 esiteltyjen vertailukohteiden perusteella. Vertailun tuloksia esitellään luvuissa 6.5.1–6.5.7. Vertailun yhteenveto esitetään luvussa 6.6.

6.5.1 UML-mallinnuksen arviointi

Taulukossa 2 on esitelty **arvioitujen mallinnusvälineiden tukemat kaaviotyypit**. Osaa kaaviotyypeistä ei ole mallinnusvälineiden käyttöliittymätoteutuksissa esitetty omina kokonaisuuksinaan, vaan niihin liittyvät näkymäelementit on liitetty jonkin toisen kaaviotyypin yhteyteen. Esimerkiksi oliokaavion näkymäelementit on voitu sijoittaa luokkakaavion yhteyteen.

UML-kaaviotyyppi	ArgoUML	Modelio	White-StarUML	Rational Modeler
Komponenttikaavio	*	x	x	x
Käyttötapaus	x	x	x	x
Luokkakaavio	x	x	x	x
Sekvenssikaavio	x	x	x	x
Sijoittelukaavio	x	x	x	x
Tilakaavio	x	x	x	x
Aktiviteettikaavio	x	x	x	x
Oliokaavio	*	x	*	*
Pakkauskaavio	*	x	*	*
Ajoituskaavio	-	-	-	-
Kokoava vuorovaikutuskaavio	-	x	-	-
Koostekaavio	-	x	x	x
Profiilikaavio	-	-	-	-
Yhteistoimintakaavio / Yhteistyökaavio	-/x	x/-	-/x	-/x

Taulukko 2: Mallinnusvälineiden tukemat UML-kaaviotyypit (x = tuettu, * = toisen kaaviotyypin yhteydessä, - = ei tuettu).

Liitteessä 1 on esitetty **ohjelmistojen tuki luokka-, sekvenssi-, aktiviteetti-, tila- ja käyttötapauskaavioiden merkintätavoille**. Vaikka merkintätapojen arviointikohteet on pyritty muodostamaan mahdollisimman yksiselitteisiksi, vaatii niiden tulkinta ohjelmistojen erilaisista toteutuksista johtuen käytännössä tapauskohtaista harkintaa. Arvioinnissa onkin päädytty käyttämään arvojen *toteutuu* ja *ei toteudu* lisäksi arvoa *toteutuu osin*. Useimmiten osittainen toteutus tarkoittaa käytännössä sitä, että mallinnuselementtiin on mahdollista liittää haluttu informaatio, mutta sen graafinen esittäminen kaaviossa ei ole mahdollista.

Joissain tapauksissa puutteellisesti tuettu ominaisuus on helposti kierrettävissä. Esimerkiksi Modeliossa aktiivinen luokka olisi mahdollista esittää myös graafisesti muokkaamalla näkymäelementin ulkoasua. Koska näkymäelementin ulkoasun muutos ei ole kytköksissä mallinnuselementin ominaisuuden muuttumiseen, ei ominaisuuden ole katsottu olevan täysin tuettu.

Luvussa 3.3.1 esitellyn **luokkakaavion** merkintätavat ovat kaikissa ohjelmistoissa hyvin tuettuja. Modelio-ohjelmistossa lähes kaikkien tarkasteltujen merkintätapojen esittäminen kaaviossa oli mahdollista. Eniten puutteita oli ArgoUML-ohjelmistossa.

Aktiviteetti-, tila- ja sekvenssikaavioiden luvuissa 3.3.4, 3.3.5 ja 3.3.7 kuvattujen merkintätapojen esittämistä tukivat parhaiten Modelio ja Rhapsody Modeler. Vähiten merkintätapoja oli käytettävissä ArgoUML:ssa. Kaavioiden perusmerkinnät on mahdollista esittää kaikilla ohjelmistoilla.

Luvussa 3.3.6 esitellyn **käyttötapauskaavion** merkintöjen piirtäminen on mahdollista kaikilla ohjelmistoilla. Ainoa havaittu poikkeama on ArgoUML:sta puuttuva järjestelmän rajaa osoittava näkymäelementti, joka on visuaalisesti korvattavissa piirtotyökalun laatikkokuviolla.

ArgoUML, Modelio ja WhiteStarUML eivät varoita, jos käyttäjä **poistaa mallista mallinnuselementin**, johon liittyy muita mallinnuselementtejä tai kaavioita. On esimerkiksi mahdollista poistaa käyttötapauskaaviosta käyttötapaus, johon on kytketty sekvenssikaavio. Tällöin myös sekvenssikaavio poistetaan mallista. Modelio poistaa mallinnuselementin aina myös mallista, kun taas ArgoUML- ja WhiteStarUML-ohjelmistoissa on erilliset

toiminnot kaaviosta ja mallista poistamista varten. Rhapsody Modeler varmistaa käyttäjältä, poistetaanko mallinnuselementti mallista, mutta poistaa tämän jälkeen elementin ilmentymät muistakin kuin näkyvillä olevasta kaaviosta erikseen varoittamatta. Myös mallinnuselementtiin liittyvät kaaviot poistetaan ilman varoitusta. Näkymäelementin poistamista kaavioista ei varmisteta.

Kaikissa vertailluissa ohjelmistoissa oli mahdollista **tallentaa kaavioita kuvina**. Mahdollisuutta mallin tallentamiseen PDF- tai HTML-muodossa ei ollut käytettävissä yhdessäkään ohjelmistossa. Tallennusmahdollisuus BMP-, JPG- ja PNG-tiedostomuodoille on esitetty taulukossa 3 ja yhteenveto kaikista tiedostomuodoista liitteessä 4. Kuvina tallennetut kaaviot vastasivat pääosin hyvin mallinnusvälineessä piirrettyä kuvaa. ArgoUML:sta EPS-muotoon tallennetusta kaaviosta puuttui assosiaatioviivoja. Modeliossa tallennus GIF-muotoon ei toiminut lainkaan.

Tiedosto-muoto	ArgoUML	Modelio	White-StarUML	Rational Modeler
BMP	-	x	x	x
JPG	-	x	x	x
PNG	x	x	-	-
PDF	-	-	-	-
HTML	-	-	-	-

Taulukko 3: Ohjelmistojen tuki kaavioiden tallentamiselle: x = tuettu, - = ei tuettu.

Kaikissa mallinnusvälineissä oli mahdollista sovittaa tulostettava kaavio arkille. WhiteStarUML mahdollisti lisäksi useiden kaavioiden sovittamisen samalle arkille. Tulostuksen esikatselu oli käytettävissä kaikissa mallinnusvälineissä ArgoUML:ää lukuun ottamatta.

Jokainen arvioiduista mallinnusvälineistä tuki **ristiriitojen tarkastamista ja mallin kritisointia**, joskin esitystavassa ja toiminnassa oli eroja. Pintapuolisessa tarkastelussa huomiota herätti se, että WhiteStarUML:ssä käyttäjän oli käynnistettävä toiminto erikseen, kun se muissa toimi automaattisesti.

6.5.2 Käytettävyyden arviointi

Vertailtujen ohjelmistojen käytettävyydessä oli selviä eroja. Jokaisessa niistä oli omat puutteensa ja etunsa. Ohjelmistojen käytettävyydestä tehdyt havainnot on esitelty luvuissa 6.5.3–6.5.6. Käytettävyyteen vaikuttavien toimintojen lista havaintoineen on liitteessä 2 ja luvun 4.2 heuristiikkaan liittyvät havainnot on esitetty liitteessä 3.

Luvussa 4.2 esitetyllä asteikolla katastrofaalisia **käytön estäviä ongelmia** ei havaittu lukuunottamatta WhiteStarUML:n päätymistä tilaan, jossa työskentely oli mahdotonta toistuvien virheilmoitusten vuoksi. Isoja ongelmia havaittiin muutamia.

Useampi arvioija olisi todennäköisesti pystynyt muodostamaan monipuolisemman näkemyksen mallinnusvälineiden käytettävyydestä. Tekijän yksin suorittama arviointi on kuitenkin jo sellaisenaan suuntaa-antava, ja sitä voi käyttää tukena mallinnusvälinettä valittaessa. Koska varsinkin tyytyväisyyden arviointi on väistämättä subjektiivista, on käyttäjän järkevää kokeilla mahdollisia ohjelmistoja itse ennen lopullisen valinnan suorittamista, mikäli tämä on mahdollista. Esimerkkisovelluksen mallia laadittaessa arvioija havaitsi jokaisen ohjelmiston käytettävyydessä heikkouksia, jotka vähensivät tyytyväisyyttä ohjelmistoon.

6.5.3 ArgoUML:n käytettävyyden arviointi

ArgoUML:n käyttöliittymä vaikuttaa ensi näkemällä karsitulta, ja sen ulkoasu poikkeaa Windows-ympäristössä totutusta. Käyttöliittymän osien merkitys vaikuttaa kuitenkin selkeältä.

Työskentelyn aloittaminen on nopeaa, koska uuden mallin laatimisen voi aloittaa suoraan ohjelman käynnistyttyä. Perustoimintojen käytön oppiminen on vaivatonta. Monien toimintojen käyttö kuitenkin vaatii kokeilua, ja ainakin arvioija havaitsi joitakin toimintoja sattumalta.

Liitteessä 2 luetelluista toiminnoista ArgoUML:ssa on käytettävissä vain osa. Työskenneltäessä erityisesti peruutustoiminnon puuttuminen häiritsi mallin laatimista. Puute liittyy luvussa 4.2 esitellyn heuristiikan kohtaan: *Ohjelmassa ja sen eri osissa tulee olla selkeät*

poistumisreitit. Puutteella on merkittävää vaikutusta käytettävyyteen, joten luvun 4.3 asteikolla se on iso ongelma.

Valitun näkymäelementin tietojen näyttäminen ohjelmaikkunan alareunassa vaikutti toimivalta ratkaisulta. Samaa aluetta käytetään kuitenkin myös mallinnuselementin piirteiden tarkempaan määrittämiseen, jolloin käyttäjältä saattaa kadota käsitys muokattavasta kohteesta.

ArgoUML:n käyttäjälle antama palaute on vähäistä. Esimerkiksi yritettäessä piirtää sääntöjen vastainen yhteys kahden näkymäelementin välille, käyttäjälle ei anneta mitään palautetta, vaan yhteys jätetään piirtämättä. Tämä on poikkeama suhteessa luvun 4.2 heuristiikan periaatteeseen *järjestelmän tulee antaa käyttäjälle kunnollista ja riittävän nopeaa palautetta*, jonka lisäksi vuorovaikutuksessa ei hyödynnetä hiiren osoittimen muuttamista.

ArgoUML-ohjelmiston Source Forge -sivulla ei ilmoiteta keskustelupalstan osoitetta, mutta sen omalla sivulla viittaus on. Keskustelupalstalla [argouml-users.net] oli kuukauden aikana esitetty kolme kysymystä, joihin yhteenkään ei ollut vastattu. Ohjelmiston mukana toimitetaan käyttöohje, muttei esimerkkikaavioita tai -malleja.

6.5.4 Modelion käytettävyyden arviointi

Modelion käytettävyydessä on joitain isoja puutteita, vaikka arvioija oli siihen kokonaisuutena tarkastellen tyytyväinen. Kuten liitteen 2 taulukosta käy ilmi, Modeliossa ei muista mallinnusvälineistä poiketen ole mahdollista poistaa pelkästään mallinnuselementtiä vastaavaa näkymäelementtiä yksittäisestä kaaviosta. Poistotoiminto poistaa myös mallinnuselementin koko mallista.

Kopioi-, leikkaa- ja liitätoiminnot ovat käytettävissä vain tietyissä näkymissä. Nämä puutteet voi nähdä poikkeamana luvun 4.2 heuristiikan periaatteesta *käyttöliittymän tulee olla yhdenmukainen*. Hakutoiminnon puuttuminen puolestaan heikentää käytön tehokkuutta, ja sen voi katsoa olevan ristiriidassa ainakin periaatteen *käyttäjän muistikuormitus tulee minimoida* kanssa.

Uuden projektin luomiseen liittyvät määritykset on Modeliossa jaettu kolmelle välilehdelle. Ensimmäisillä käyttökerroilla oletusnäkymä vaikutti täydeltä useiden ikkunoiden vuoksi. Monia toimintoja ei voi käynnistää ohjelmiston päävalikosta, vaan ne on valittava tilannekohtaisesta pikavalikosta.

Modelion piirtotoiminnossa havaittiin esimerkkiohjelmistoa mallinnettaessa vikoja. Esimerkiksi yhteysviiva saattoi siirtyä muokattaessa automaattisesti, eikä sitä ollut tämän jälkeen mahdollista siirtää takaisin alkuperäiseen kohtaan. Myöskään peruutustoiminnon käyttäminen ei vaikuttanut viivan sijaintiin. Tämän voi arvioida olevan luvussa 4.3 esitetyllä asteikolla iso käytettävyysongelma.

Piirtotoiminnon käytettävyyttä heikensi myös ajoittain esiintynyt valinnan epätarkkuus, jonka seurauksena tietyn näkymäelementin valinta saattoi vaatia useita yrityksiä. Tämä vähensi käytön tehokkuutta samoin kuin se, että mallinnus- ja näkymäelementtejä ei voi muokata monivalinnassa.

Modelio antaa käyttäjälle selkeästi palautetta muuttamalla hiiren osoitinta ja kaavion osien värejä tilanteesta riippuen. Tältä osin ohjelman käyttöliittymä on selkeästi muita vertailtuja ohjelmistoja parempi.

Modelion keskustelupalstalla [Modelio Community 2013b] oli kuukauden aikana avattu seitsemän kysymyksen sisältänyttä keskustelua. Näistä viiteen oli vastattu vähintään kerran. Ohjelmiston mukana toimitetaan käyttöohje, mutta ei esimerkkikaavioita tai -malleja.

6.5.5 WhiteStraUML:n käytettävyyden arviointi

WhiteStarUML-ohjelmisto näyttää Windows-käyttäjälle ulkoasultaan ja toiminnoiltaan helposti omaksuttavalta. Siinä on käytettävissä kaikki liitteessä 2 luetellut toiminnot. Ohjelmisto on beta-vaiheessa, ja siinä havaittiin koekäytön aikana yksi katastrofaaliseksi katsottava virhetilanne.

Ohjelman käynnistymisen jälkeen on valittava joko yksi viidestä tavasta luoda uusi projekti tai aiemmin luodun tiedoston avaaminen. Tyhjään projektiin käyttäjän on luotava vielä uusi malli ennen kuin ensimmäisen kaavion lisääminen on mahdollista.

Kaavion piirtäminen WhiteStarUML:lla on arvioijan mielestä muutamista heikkouksista huolimatta sujuvaa. Heikkoudeksi voi mainita esimerkiksi sen, että piirrettäessä useampi samanlainen näkymäelementti on tyyppi valittava ennen kunkin näkymäelementin lisäämistä työkalupalkista.

WhiteStarUML:n käyttäjälle antamat ilmoitukset ovat epäselviä, mikä rikkoo luvun 4.2 heuristiikan periaattetta *virheilmoitusten tulee olla selkeitä ja ymmärrettäviä*. Esimerkiksi, jos yritetään piirtää sääntöjen vastainen yhteys kahden mallinnuselementin välille, virheilmoitus kehottaa osoittamaan yhteyden päätepisteet tarkemmin.

Vuorovaikutuksessa ei hyödynnetä hiiren osoittimen muuttamista. Tämä rikkoo luvun 4.2 heuristiikan periaatteita *järjestelmän tulee antaa käyttäjälle kunnollista ja riittävän nopeaa palautetta* ja *virhetilanteisiin joutumista tulisi välttää*.

Ilmeisesti käyttöliittymän mukauttamisen seurauksena WhiteStarUML päätyi tilaan, jossa projektin avaus ei näyttänyt onnistuvan, vaan keskeytyi virheilmoitukseen. Uudelleenasennuksen yhteydessä päivitetyn version asennuksen jälkeen projektin avaaminen oli jälleen mahdollista.

WhiteStarUML-projektin sivulla ei ilmoiteta keskustelupalstan tai muun tukikanavan osoitetta. Ohjelmiston mukana toimitetaan käyttöohje ja muutamia esimerkkimalleja.

6.5.6 Rhapsody Modelerin käytettävyyden arviointi

Rhapsody Modeler näyttää arvioijasta Windows-käyttäjälle ulkoasultaan ja toiminnoiltaan helposti omaksuttavalta. Esimerkiksi työkalupalkin kuvakkeet ja päävalikon toimintojen sijoittelu ovat samanlaisia kuin Windows-ympäristössä on totuttu käyttämään.

Käynnistyttyään ohjelmisto näyttää aloitusnäkymän, jossa on valittavissa kuusi toimintoa. Uutta projektia luotaessa riittää periaatteessa tarkistaa muutama projektille annettu oletustieto, kuten projektin nimi. Arviointia suoritettaessa havaittiin, että projektin luominen keskeytyy, jos projektikansion luominen ei onnistu oletuspolkuun puuttuvien käyttöoikeuksien vuoksi. Projektin määrittäminen on tällöin aloitettava alusta.

Kaavion piirtämiseen tarvittavat perustoiminnot ovat helposti omaksuttavissa, mutta osa toiminnoista on vaikeasti löydettävissä. Esimerkiksi stereotyypin ominaisuuksien muuttamiseksi käyttäjän pitää ensin osata avata `Browser`-ikkuna, joka ei ole oletuksena näkyvissä. Heikkoudeksi voi mainita myös sen, että piirrettäessä useampi samanlainen näkymäelementti on sen tyyppi valittava työkalupalkista ennen kunkin näkymäelementin lisäämistä.

Ohjelmiston antamat ilmoitukset olivat selkeitä ja ymmärrettäviä. Kaavioita piirrettäessä käyttäjälle annettiin palautetta hiiren osoitinta muuttaen muutamissa tilanteissa.

Tyytyväisyyttä vähensi se, että palattaessa aiemmin luotuun projektiin, käyttöliittymä näytti erilaiselta kuin aiemmilla käyttökerroilla. Esimerkiksi mallinnuselementtien selausikkuna saattoi jäädä avautumatta. Tämän voi katsoa olevan poikkeama suhteessa luvun 4.2 heuristiikan periaatteeseen *käyttöliittymän tulee olla yhdenmukainen*.

Rational Rhapsodyn keskustelufoorumin [IBM 2013b] viimeisin kysymys oli päivätty yli puoli vuotta tarkasteluajankohtaa ennen ja sitä edellinen yli vuosi ennen tätä viestiä. Ohjelmiston mukana ollut käyttöohje ei avautunut ohjelmiston päävalikon kautta. Asennuskansiossa se kuitenkin oli luettavissa PDF-tiedostomuodossa. Ohjelmiston mukana toimitetaan myös muutamia esimerkkimalleja.

6.5.7 Elinkaarikustannusten arviointi

Mallinnusvälineiden elinkaarikustannuksia arvioitiin asennuspaketin koon, asennusprosessin sujuvuuden ja käyttöjärjestelmätuen pohjalta. Käyttöjärjestelmätuki on esitetty taulukkona ja asennuspaketin koko kaaviona liitteessä 6.

Kaikkien ohjelmistojen **asennuspaketit** voidaan ladata Internetistä. Rhapsody Modelerin asennuspaketin lataaminen vaatii rekisteröitymisen. Modelio-ohjelmistosta latautui oletuksena 64-bittinen versio. Asennuspaketeista pienimmät olivat ArgoUML 16 Mt ja WhiteStarUML 26 Mt. Modelion asennuspaketin koko oli 179 Mt, joka oli noin kolminkertainen verrattuna toiseksi suurimpaan Rhapsody Modelerin 64 Mt asennuspakettiin.

Jokaisen vertaillun **ohjelmiston asennus** tehdään ohjatusti, ja asennuksen suorittaminen vei kaikissa tapauksissa alle 10 minuuttia. Nopeimmin asentui ArgoUML, vaikka sen asennuksen yhteydessä asennettiin Java JRE. Kaikkien ohjelmistojen asennus sujui ongelmitta. Tosin ennen Modelion asennusohjelman käynnistämistä se on purettava zip-pakkauksesta, ja WhiteStarUML:n asennus vaatii pääkäyttäjän oikeudet.

WhiteStarUML ja Rhapsody Modeler vaativat **käyttöjärjestelmän** osalta toimiakseen Windows-työaseman. Modeliosta on saatavilla käännös Linux- ja Windows-ympäristöihin. ArgoUML on ainakin periaatteessa asennettavissa mihin tahansa ympäristöön, jolle on saatavilla Java-tulkki. Lisäksi ArgoUML:ää on mahdollista käyttää ilman työasemaan tehtävää asennusta.

Kukin ohjelmisto toimi laitteistoltaan perustasoisessa Windows 7 -työasemassa. Modelion toiminnassa oli havaittavissa ajoittaista lievää hitautta, mutta ilman tarkempaa selvitystä ei voida varmuudella sanoa sen johtuneen liian vähäisistä laitteistoresursseista.

Merkittävin ohjelmistojen kokeilun aikana tehty ylläpitotyö oli WhiteStarUML:n uudelleenasennus. Myös Rational Rhapsody vaati ylläpitoa, sillä se ei pystynyt tallentamaan mallia ehdottamaansa oletuskansioon. Pidemmällä ajanjaksolla vaikutusta voi olla myös mallinnustyökalujen julkaisutiheydellä, mutta testausjakson aikana uudelleenasennus ei tästä syystä ollut tarpeen.

6.6 Suositukset ja johtopäätökset

Modelio ja WhiteStarUML ovat kokonaisuutena arvioiden vertailukohtana käytetyn Rhapsody Modelerin veroisia, mutta kaikissa näissä kolmessa on omat puutteensa ja etunsa. ArgoUML on arvioiduista ohjelmistoista heikoin, vaikka sekin lienee useimmiten riittävä esimerkiksi luokka- ja sekvenssikaavioiden piirtämiseen.

Modelio oli Rational Rhapsodyyn verrattuna tasavertainen tuettujen UML-kaavioiden ja merkintätapojen osalta. Selvästi heikoimmin tässä suhteessa menestyi ArgoUML, jonka puutteet UML 2 -versioiden tuen osalta olivat tiedossa jo dokumentaation perusteella.

Käytettävyydeltään arvioijan mielestä kokonaisuutena parhaita olivat WhiteStarUML ja Rhapsody Modeler. WhiteStarUML tosin jouduttiin asentamaan koekäytön aikana kerran uudelleen ohjelmiston päädyttyä käytön estävään virhetilaan. Modeliossa ja ArgoUML:ssa on muutamia isoja merkittävästi käytettävyyteen vaikuttavia ongelmia, mutta toisaalta joitain hyviä puolia. Varsinkin Modelion tapa antaa käyttäjälle informaatiota hiiren osoittimen ulkonäköä muuttamalla, oli muihin vertailtuihin ohjelmistoihin verrattuna erinomainen.

ArgoUML on käytettävyydeltään muita arvioituja mallinnustyökaluja heikompi. Esimerkiksi peruutustoiminnon puuttuminen on merkittävä heikkous. Toisaalta mallintamisen aloittaminen on ArgoUML:ssa kaikkein nopeinta.

Elinkaarikustannusten osalta ohjelmistojen välillä ei vaikuta olevan suurta eroa. Jokaisen ohjelmiston asentaminen on tehtävissä noin kymmenessä minuutissa. WhiteStarUML:n asennus tosin vaatii pääkäyttäjän oikeudet, mikä voi joissain tapauksissa olla ongelma. ArgoUML:n asennuspaketti on tiedostokooltaan kaikkein pienin ja Modelion suurin.

Ohjelmistojen ylläpidon osalta on huomioitava ArgoUML:n vaatima Java JRE ja ainakin beta-vaiheessa olevan WhiteStarUML:n mahdollisesti tiheä päivitystahti. Elinkaarikustannuksiin liittyen ArgoUML:n etu on mahdollisuus käyttää ohjelmistoa useilla eri laitealustoilla. Myös Modelio on mahdollista asentaa Sovellusprojektit-opintojaksolla tavallisesti käytössä oleviin Windows- ja Linux-työasemiin, mutta WhiteStarUML ja Rhapsody Modeler vaativat toimiakseen Windows-käyttöjärjestelmän.

Mikäli toivotaan mahdollisimman laajaa tukea UML:n versiolle 2, on Modelio vertailuista ohjelmistoista paras valinta. WhiteStarUML on käytettävyydeltään Modeliota parempi, mikäli tyydytään UML 1.4 -versioon ja toimintaan vain Windows-alustalla.

Sekä Modelio että WhiteStarUML tulevat todennäköisesti kehittymään nykyisistä versioistaan. Niitä molempia voi edellä kuvatuin varauksin suositella käytettäväksi Sovellusprojekti-opintojaksolla. ArgoUML:n käyttö ei ole perusteltua kuin poikkeustapauksissa. Rhapsody Modelerin käyttöä kannattanee harkita, jos ohjelmistokehityksessä käytetään myös muita saman tuoteperheen ohjelmistoja.

Edellä esitetyt johtopäätökset koskevat myös ohjelmistojen käyttöä mikroyrityksissä. Niissä voi kuitenkin valintahetkellä olla tarpeen arvioida ohjelmistojen kehitystahtia ja kehityksen jatkuvuutta tarkemmin kuin kestoltaan rajatuissa Sovellusprojekteissa.

7 Yhteenveto

Tutkielman tavoitteena oli määrittää avoimen lähdekoodin UML-mallinnustyökalujen arviointiin soveltuvat vertailukohteet siten, että niitä käyttäen voidaan arvioida ohjelmistojen soveltuvuutta käytettäväksi Sovellusprojekti-opintojaksolla ja mahdollisesti samalla mikroyrityksissä. Vertailukohteet jaettiin kolmeen ryhmään, joista keskeisimmät olivat tuki UML-mallin laatimiselle ja käytettävyys. Lisäksi arvioitiin pinnallisemmin ohjelmistojen elinkaarikustannuksia.

Määritettyjen vertailukohteiden avulla oli mahdollista havaita eroja mallinnustyökalujen välillä, sekä arvioida niiden soveltuvuutta mainittuihin käyttötarkoituksiin. Jotkin vertailukohteista osoittautuivat käytännössä epätäsmällisiksi lähinnä mallinnusvälineiden toisistaan poikkeavien toteutusten vuoksi. Niiden tarkentaminen ei kuitenkaan olisi välttämättä parantanut arvioinnin täsmällisyyttä. Lisäksi yksityiskohtaisempien vertailukohteiden soveltuvuutta muihin kuin tutkielmassa tarkasteltuihin mallinnusvälineisiin on myös mahdotonta arvioida.

Vertailtaviksi avoimen lähdekoodin ohjelmistoiksi valitut ohjelmistot ArgoUML ja WhiteStarUML tukevat UML:n versiota 1. Vertailukohtana käytetyn Rational Rhapsody -ohjelmiston lisäksi Modelio tukee UML:n 2-versiota. Se olikin vertailluista ohjelmistoista paras UML-tuen osalta. Toisaalta ajatellussa käyttötarkoituksessa tuki uusimmalle UML-versiolle ei ole ehdoton edellytys.

Käytettävyyden arviointi perustuu Nielsenin listaan, jonka mukaisesti tutkielman tekijä suoritti arvioinnin. Arviointi on sellaisenaan suuntaa-antava, ja sitä voi käyttää tukena mallinnusvälinettä valittaessa. Koska varsinkin tyytyväisyyden arviointi on väistämättä subjektiivista, on käyttäjän joka tapauksessa syytä kokeilla mahdollisia ohjelmistoja itse ennen lopullisen valinnan suorittamista.

Käytettävyyttä arvioitaessa huomioitiin myös luvussa 4.3 esitellyn asteikon ulkopuolelle jääneet positiiviset huomiot. Ne ovat vertailun kannalta samalla tavalla merkittäviä kuin käytettävyyden puutteetkin, joten ne huomioitiin arviointiasteikkoa määritettäessä. Nielse-

nin asteikko sopii vertailua paremmin käytettäväksi yksittäisen ohjelmiston käytettävyydestä tehtyjen havaintojen luokittelussa.

Käytettävyydeltään vertaillut mallinnusvälineet olivat ArgoUML:ää lukuunottamatta tasaveroisia. Kaikissa vertailluissa ohjelmistoissa havaittiin käytettävyyteen liittyviä puutteita, mutta näistä vain yksi WhiteStarUML:n puute oli käyttöä estävä.

Elinkaarikustannuksiltaan ohjelmistot ovat melko tasavertaisia, vaikka jotain eroja ohjelmistojen välillä on. Esimerkiksi Modelion asennusohjelma on ennen asennusta purettava zip-pakkauksesta, ja WhiteStarUML:n asentaminen vaatii pääkäyttäjän oikeudet. Lisäksi beta-vaiheessa oleva WhiteStarUML jouduttiin asentamaan uudelleen testijakson aikana.

Arvioinnin perusteella avoimen lähdekoodin ohjelmistoista Modelio ja WhiteStarUML soveltuvat puutteistaan huolimatta käytettäväksi Sovellusprojektit-opintojaksolla. Suljetun lähdekoodin Rhapsody Modeler -ohjelmiston kokeiluversio ei niihin verrattuna ollut kokonaisuutena parempi. Muutamista hyvistä ominaisuuksistaan huolimatta ArgoUML oli tämän vertailuin heikoin ohjelmisto.

Tutkielmassa ei tarkasteltu mallinnusohjelmistoja osana laajempaa ohjelmistokehitysvälineiden muodostamaa kokonaisuutta. Tarjolla olevat kehitysympäristöt saattaisivat olla mielenkiintoinen tutkielman kohde, sillä niitä käytettäessä mallia voitaisiin hyödyntää suoraan osana ohjelmistokehitysprosessin tehtäväkokonaisuuksia, sekä mallia voitaisiin ylläpitää samassa kehitysympäristössä.

Mallinnusvälineiden toimivuutta eri käyttöjärjestelmäympäristöissä voisi myös olla tarpeen tarkastella. Tällöin olisi myös mahdollista selvittää, onko useisiin käyttöjärjestelmäympäristöihin saatavilla olevien mallinnusvälineiden toiminnassa käyttöjärjestelmästä tai käännöksestä johtuvia eroja.

Lähteet

Auer M., Tschurtschenthaler T. and Biffl S., *A Flyweight UML Modeling Tool for Software Development in Heterogeneous Environments*, Vienna University of Technology, Institute of Software Technology, 2003.

argouml-users.net, *Board index*, saatavana HTML-muodossa <URL: `http://argouml-users.net/forum/`>, viitattu 2.8.2013.

Barnum Carol M., "Usability Testing Essentials", Elseviewer Inc., 2011.

Booch Grady, Rumbaugh James and Jacobson Ivar, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.

Borcoman Ionutz , *PlantUML QEditor*, saatavana HTML-muodossa <URL: `http://sourceforge.net/projects/plantumlqeditor/`>, viitattu 27.4.2013.

de Freitas Leonardo Guilherme and Nikolai, *DoUML*, saatavana HTML-muodossa <URL: `http://sourceforge.net/projects/douml/`>, viitattu 25.4.2013.

Dobing Brian, Parsons Jeffrey, *Dimensions of UML Diagram Use: A Survey of Practitioners*, Journal of Database Management, Volume 19, Issue 1, 2008.

Eichelberger Holger, Eldogan Yilmaz and Schmid Klaus, *A Comprehensive Survey of UML Compliance in Current Modelling Tools*, University of Hildesheim, Gesellschaft für Informatik, 2009.

Eriksson Hans-Erik and Penker Magnus, "UML", Oy Edita Ab, Jyväskylä, 2000.

Euroopan yhteisöjen komissio, *Mikroyritysten sekä pienten ja keskisuurten yritysten määritelmästä*, saatavana PDF-muodossa <URL: `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2003:124:0036:0041:FI:PDF`>, Euroopan unionin virallinen lehti, 20.5.2003.

Fowler Martin, "UML Distilled: A Brief Guide to the Standard Object Modeling Language", Third Edition, Addison-Wesley, 2004.

Free Software Foundation, *Philosophy of the GNU Project*, saatavilla HTML-muodossa <URL: http://www.gnu.org/philosophy/philosophy.html>, viitattu 20.10.2013.

Gaphor team, *Gaphor UML Modeling*, saatavana PHP-muodossa <URL: http://gaphor.sourceforge.net/features.php>, viitattu 27.4.2013.

Gilbert John, *Taylor, Model Driven Architecture on Rails*, saatavana HTML-muodossa <URL: http://taylor.sourceforge.net/index.php/Overview>, 19.10.2013.

Heikkilä Hilkka, Inkeroinen Markus, Isomöttönen Ville, Kannisto Lari, Korhonen Vesa, Kärkkäinen Kari ja Santanen Jukka-Pekka, *Toteutettuja Tietotekniikan Sovellusprojekteja*, saatavilla HTML-muodossa < URL: http://www.mit.jyu.fi/opiskelu/sovellusprojektit/toteutetut.html>, Jyväskylän yliopisto, tietotekniikan laitos, 25.4.2013.

Hirzel Peter, "UML-Editor Reference Manual", saatavana PDF-muodossa <URL: http://www.umleditor.org/download/refman_en.pdf>, 6.2.2004.

Horstmann Cay S. and de Pellegrin Alexandre, *Violet UML Editor*, saatavana HTML-muodossa <URL: http://violet.sourceforge.net/>, viitattu 27.4.2013.

IBM, "Evaluate: IBM Rational Modeler", saatavana HTML-muodossa <URL: http://www.ibm.com/developerworks/downloads/r/modeler/>, viitattu 25.4.2013.

IBM, Rational Modeler Forums, saatavana HTML-muodossa <URL: https://www.ibm.com/developerworks/community/forums/html/forum?id=11111111-0000-0000-0000-000000001511>, viitattu 2.8.2013.

Khaled Lena, *A Comparison between UML Tools*, ICECS '09, Second International Conference on Environmental and Computer Science, 2009.

Kirchner Lutz und Jung Jürgen, *Ein Bezugsrahmen zur evaluierung von UML-modellierungswerkzeugen*, Institut für Wirtschatsinformatik, Universität Koblenz-Landau, 2001.

Kollmann Ralf, Selonen Petri, Stroulia Eleni, Systä Tarja and Zündorf Albert, *A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering*, Proceedings of the Ninth Working Conference on Reverse Engineering, IEEE Computer Society, 2002.

Koskimies Kai, Koskinen Johannes, Maunumaa Mika, Peltonen Jari, Selonen Petri, Siikarla Mika ja Systä Tarja, *UML työvälineenä ja tutkimuskohteena*, saatavilla pdf-muodossa <URL: http://www.cse.tkk.fi/fi/tkt-lehti/a21/uml.pdf>, SYTYKE ry, Tietojenkäsittelytiede, Numero 21, syyskuu 2004.

Lakhani Karim R. and Hippel Eric von, *How Open Source Software Works: "Free" User-to-user Assistance*, MIT Sloan School of Management, 12.11.2001.

Modelio Community, *Modelio - Modeling environment (UML)*, saatavilla HTML-muodossa <URL: http://sourceforge.net/projects/modeliouml/>, viitattu 27.4.2013.

Modelio Community, *Modelio forum*, saatavilla HTML-muodossa <URL: http://www.modelio.org/forum/>, viitattu 2.8.2013.

Nielsen Jakob, "Usability Engineering", Academic Press Inc., 1993.

OMG, "Diagram Definition (DD), Version 0.9", saatavilla pdf-muodossa < URL: http://www.omg.org/spec/DD/1.0/Source/10-05-01.pdf>, 24.5.2010.

OMG, "OMG Unified Modeling Language (OMG UML), Infrastructure 2.4.1", saatavilla pdf-muodossa <URL: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF>, 5.8.2011.

OMG, "OMG Unified Modeling Language (OMG UML), Superstructure 2.4.1", saatavilla PDF-muodossa <URL: http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF>, 6.8.2011.

Open Source Initiative, *The Open Source Definition*, saatavilla HTML-muodossa <URL: http://www.opensource.org/>, viitattu 15.4.2013.

Quesenbery Whitney, *Dimensions of Usability: Defining the Conversation, Driving the Process*, saatavilla pdf-muodossa <URL: http://www.wqusability.com/articles/5es-upa2003.pdf>, Proceedings of the UPA 2003 Conference, June 23–27, 2003.

Rautiainen Juha, "UML-mallinnuskieli ja sen hyödyntäminen ohjelmistokehityksessä", LuK-tutkielma, Jyväskylän yliopisto, tietotekniikan laitos, 13.3.2011.

Rumbaugh James, Jacobson Ivar and Booch Grady, "The Unified Modeling Language Reference Manual", Addison Wesley Longman Inc., 1999.

Rumbe Bernhard, *Executable Modeling with UML - A Vision or a Nightmare?*, saatavilla PDF-muodossa <URL: http://dml.wdfiles.com/local--files/papers:executable-modeling-with-uml-a-vision-of-a-nightmare/Executable%20Modeling%20With%20Uml%20A%20Vision%20Of%20A%20Nightmare.pdf>, Munich University of Technology, viitattu 21.10.2013.

Santanen Jukka-Pekka, *Tietotekniikan Sovellusprojekteja Jyväskylän yliopistossa*, saatavissa HTML-muodossa <URL: http://www.mit.jyu.fi/opetus/sovellusprojektit/>, Jyväskylän yliopisto, tietotekniikan laitos, 11.12.2008.

Santanen Jukka-Pekka, *Tietotekniikan Sovellusprojektien ohje*, saatavissa PDF-muodossa <URL: http://www.mit.jyu.fi/palvelut/sovellusprojektit /projohje.pdf>, Jyväskylän yliopisto, tietotekniikan laitos, 29.1.2013.

Sommerville Ian, "Software Engineering", Eighth Edition, Addison-Wesley, 2007.

Szpilewski Janusz and Zuurbier Albert, "WhiteStarUML", saatavana pdf-muodossa <URL: http://sourceforge.net/projects/whitestaruml/files/Docs-1.1.zip/download>, 2013.

Szpilewski Janusz, *WhiteStarUML*, saatavana HTML-muodossa <URL: http://sourceforge.net/projects/whitestaruml/>, viitattu 27.4.2013.

Tigris.org, *Welcome to ArgoUML*, saatavana HTML-muodossa <URL: http://argouml.tigris.org/>, viitattu 24.5.2013.

Tilastokeskus, *Suomen virallinen tilasto (SVT), Informaatiopalvelujen tilinpäätöstilasto*, saatavissa <URL: http://tilastokeskus.fi/til/iptp/index.html>, viitattu 16.4.2013.

Umbrello Project, *Umbrello UML Modeller*, saatavana HTML-muodossa <URL: http://uml.sourceforge.net/>, viitattu 27.4.2013.

Liitteet

Liite 1 UML-tuki

Mallinnusvälineiden UML-tukeen liittyvien ominaisuuksien arvioinnissa on apuna käytetty oheisia kuvaustapojen tarkistuslistoja. Taulukoihin on kunkin näkymäelementin ja merkintätavan kohdalle merkitty, onko sen käyttäminen mallinnusvälineessä mahdollista (x), osin mahdollista (*) tai mahdotonta (-).

Luokkakaavio				
Näkymäelementit ja merkintätavat	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Luokka	x	x	x	x
Osioiden piilottaminen	x	x	x	x
Abstraktiluokka	x	x	x	-
Ominaisuusosio	x	x	x	x
Johdetut ominaisuudet	-	x	-	-
Näkyvyyden esittäminen	*	x	x	x
Ominaisuuksien kerrannaisuus	*	x	x	x
Assosiaatio	x	x	x	x
Suunnatut assosiaatiot	x	x	x	x
Yhteysluokka	x	x	x	x
Valitsinassosiaatio	-	x	x	x
Staattiset toiminnot ja ominaisuudet	*/-	x	-	x
Toiminto-osio	x	x	x	x
Yleistyssuhde	x	x	x	x
Yleistys puumuodossa	-	x	-	-
Riippuvuussuhde	x	x	x	x
Rajoitteet	*	x	*	x
Lisäosio	-	x	x	x
Kooste ja vahva kooste	x	x	x	x
Liittymä (rajapinta)	-	x	x	x
Abstrakti toiminto	x	x	x	x
Mallinneluokka	-	x	x	x
Enumeraatio	*	x	x	-
Aktiivinen luokka	*	*	x	x

Sekvenssikaavio				
Näkymäelementit ja merkintätavat	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Osallistuja	x	x	x	x
Elämänviiva	x	x	x	x
Viesti	x	x	x	x
Löydetty viesti	-	x	-	x
Kadotettu viesti	-	x	-	x
Synkroninen ja asynkroninen kutsu	x	x	x	x
Rekursio	x	x	x	x
Paluuviesti	x	x	x	x
Osallistujan aktiivinen jakso	x	x	x	x
Parametrisoitu kutsu	-	*	x	x
Osallistujan luominen	x	x	x	x
Osallistujan tuhoaminen	x	x	x	x
Lohko	-	x	x	x
Silmukka	-	x	x	x
Ehto	-	x	x	x

Aktiviteettikaavio				
Näkymäelementit ja merkintätavat	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Toimenpide	x	x	x	x
Siirtymä	x	x	x	x
Alkutila	x	x	x	x
Lopputila	x	x	x	x
Vyöhykkeet	x	x	x	x
Haarautumis- ja liittymispalkki	x	x	x	x
Liittymissääntö	-	*	-	-
Päätös- ja yhdistymiselementti	-	x	x	x
Kaavion osittaminen	-	x	-	x
Saapuva ja lähtevä signaali	*	x	x	x
Saapuva aikasignaali	*	x	-	x
Tappi (pin)	-	x	-	x
Lisäalue	-	x	-	x
Tietovuon loppu	-	x	x	-

Tilakaavio				
Näkymäelementit ja merkintätavat	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Tila	x	x	x	x
Siirtymä	x	x	x	x
Alkutila	x	x	x	x
Lopputila	x	x	x	x
Toimet	x	x	x	x
Aktiviteetti	x	x	x	x
Yli- ja alitila	x	x	x	x
Samanaikaiset tilat	x	x	-	x
Siirtymän toimintolauseke	*	x	x	x
Siirtymän varmuusehto	*	x	-	x
Siirtymän heräte	x	x	x	x

Käyttötapauskaavio				
Näkymäelementit ja merkintätavat	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Toimija	x	x	x	x
Käyttötapaus	x	x	x	x
Yhteys	x	x	x	x
Järjestelmän rajaus	-	x	x	x
Käyttötapausten välinen yhteys	x	x	x	x
Käyttötapauksen kirjallinen kuvaus	x	x	x	x

Liite 2 Ominaisuuksien tarkastuslista

Mallinnusvälineiden käytettävyyttä arvioitaessa on tukena käytetty oheista ominaisuuksien tarkastuslistaa. Taulukoihin on kunkin näkymäelementin ja merkintätavan kohdalle merkitty, onko sen käyttäminen mallinnusvälineessä mahdollista (x), osin mahdollista (*) tai mahdotonta (-).

Tarkastettava ominaisuus	**ArgoUML**	**Modelio**	**WhiteStar'UML**	**Rational**
Kopiointi	-	x	x	x
Leikkaa	-	x	x	x
Poisto kaaviosta / mallista	x	-/x	x	x
Liitä	-	x	x	x
Peruuta / tee uudelleen	-	x	x	x
Monivalinta	x	x	x	x
Pikavalikko	x	x	x	x
Työkaluvihje	x	x	x	x
Pikanäppäimet	-	x	x	x
Kaavionäkymän suurentaminen ja pienentäminen	x	x	x	x
Työtilan mukauttaminen	x	x	x	x
Etsi	x	-	x	x
Näkymäelementtien selaaminen	x	x	x	x
Usean elementin ominaisuuksien muuttaminen samanaikaisesti	-	-	x	x
Näkymäelementin lisääminen ilman ominaisuuksien määrittämistä	x	x	x	x
Näkymäelementtien tasaaminen	x	x	x	x
Näkymäelementtien koon muuttaminen	x	x	x	x
Näkymäelementtien ulkoasun muuttaminen	x	x	x	x
Usean näkymäelementin ulkoasun muuttaminen samanaikaisesti	x	*	x	x

Liite 3 Mallinnusvälineiden käytettävyydestä tehdyt havainnot

Liitteen taulukoissa esitetään mallinnusvälineiden käytettävyyspuutteista ja poikkeuksellisen hyvistä ratkaisuista tehdyt havainnot kytkettynä luvun 4.1 sääntölistaan:

a. Käyttäjän ja järjestelmän vuorovaikutuksen tulee olla yksinkertaista ja luonnollista.
b. Järjestelmän tulee käyttää käyttäjän kieltä.
c. Käyttäjän muistikuormitus tulee minimoida.
d. Käyttöliittymän tulee olla yhdenmukainen.
e. Järjestelmän tulee antaa käyttäjälle kunnollista ja riittävän nopeaa palautetta.
f. Ohjelmassa ja sen eri osissa tulee olla selkeät poistumisreitit.
g. Käyttäjälle tulee tarjota oikopolkuja.
h. Virheilmoitusten tulee olla selkeitä ja ymmärrettäviä.
i. Virhetilanteisiin joutumista tulisi välttää.
j. Käyttäjän saatavilla tulee olla kunnolliset opastustoiminnot ja dokumentaatio.

Havaintojen vakavuus on ilmaistu seuraavalla asteikolla:

0. Kyseessä ei ole käytettävyysongelma.
1. Kosmeettisella ongelmalla on vähäinen vaikutus käytettävyyteen.
2. Pienellä ongelmalla on vaikutusta käytettävyyteen.
3. Isolla ongelmalla on merkittävää vaikutusta käytettävyyteen.
4. Katastrofaalinen ongelma on este järjestelmän käyttämiselle.

Lisäksi on käytetty merkintää * ilmaisemaan myönteistä havaintoa.

ArgoUML:n käytettävyydestä tehdyt havainnot										
Havainto	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Uusi projekti voidaan aloittaa heti ohjelman käynnistyttyä.	*									
Työtilassa ei näytä olevan esillä tarpeettomia toimintoja.	*									
Ei peruutustoimintoa.						4				
Ohjeessa on puutteita.										3
Monivalinnassa vain ensimmäisen elementin tiedot ovat näkyvissä.	2									
Ominaisuuksien käsittelyyn käytetyssä ali-ikkunassa voidaan siirtyä kohteesta toiseen ilman, että käyttäjä saa tästä palautetta.			2							
Leikkaa- ja kopiotoiminnot puuttuvat.			2							
Ominaisuuksien muokkausikkunassa siirtyminen ylemmälle tasolle onnistuu vain yhdellä välilehdellä.						2				
Piirrettäessä useita samoja elementtejä on piirrettävä elementti valittava aina uudelleen.							2			
Monen elementin tietojen muuttaminen yhtä aikaa ei ole mahdollista.							2			
Stereotyypin ominaisuuksien muuttamiseksi se on avattava Browser-ikkunan kautta.							2			
Ei anneta varoitusta, jos poistettavaan näkymäelementtiin liittyy kaavioita.									2	
Ali-ikkunoiden järjestystä ei voi muuttaa.		1								
Käyttäjälle ei anneta palautetta, mikäli yritetään piirtää näkymäelementti tai assosiaatio, joka ei ole mahdollinen.					1					
Monivalinnan purkamiseksi on valittava valinnan ulkopuolinen elementti.						1				

Modelion käytettävyydestä tehdyt havainnot										
Havainto	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Värejä ja hiirenosoitinta käytetään palautteen antamiseen kattavasti.					*					
Ohje vaikuttaa kattavalta.										*
Tutoriaali on saatavilla videona.										*
Kopioi- ja leikkaatoiminnot eivät ole käytettävissä kaikissa ali-ikkunoissa.				3						
Peruuta-toiminto ei toimi kaikissa tilanteissa.					3					
Hakutoiminto ei löydä mallinnuselementtejä.						3				
Valinta on epätarkka, minkä vuoksi toimenpiteen kohdistaminen haluttuun elementtiin on ajoittain vaikeaa.	2									
Päävalikosta puuttuu toimintoja, joita siellä olettaisi olevan.			2							
Näkymäelementtiä ei voida kopioida valitsemalla kaaviosta.			2							
Otsikoiden siirtäminen saattaa olla estetty ilman näkyvää syytä ja olla myöhemmin mahdollista.				2						
Piirrettäessä useita samoja elementtejä on piirrettävä elementti valittava aina uudelleen.						2				
Elementtien tietojen muuttaminen ei ole mahdollista monivalinnassa.						2				
Ei anneta varoitusta, jos poistettavaan näkymäelementtiin liittyy kaavioita.									2	
Hakutoiminto lakkasi toimimasta ilman näkyvää syytä.									2	
Kaavion tallentaminen GIF-tiedostomuodossa ei toimi.									2	
Työtila on täysi. Esillä on paljon toimintoja, joista osa vaikuttaa päällekkäisiltä.	1									
Uuden projektin luominen on jaettu kolmelle välilehdelle.	1									
Näkymäelementin sijoittaminen esillä olevan alueen ulkopuolelle on hankalaa.	1									
Mallinnuselementin ulkoasua ja ominaisuuksia muokataan eri ikkunoissa, jotka on sijoitettu oletuksena erilleen.	1									
Muutamia mallinnuselementtejä oli sijoitettu eri ryhmään, kuin testaaja oletti.		1								

WhiteStarUML:n käytettävyydestä tehdyt havainnot										
Havainto	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Piirtäminen on sujuvaa.	*									
Piirrettäessä useita samoja näkymäelementtejä voidaan valinta lukita kaksoisnapauttamalla.							*			
Ohje ja esimerkkejä saatavilla.										*
Ohjelmisto päätyi peruuttamattomaan virhetilanteeseen.									5	
Ali-ikkuna saattaa kadota odottamatta näkyvistä.				3						
Ali-ikkunoiden uudelleen järjestäminen on vaikeaa.			2							
Näkymäelementtejä voi siirtää vain hiirellä.							2			
Epäselviä ja harhaanjohtavia virheilmoituksia. Ei viittausta ohjeeseen.								2		
Ei anneta varoitusta, jos poistettavaan näkymäelementtiin liittyy kaavioita.									2	
Projektin aloittaminen edellyttää useita valintoja.	1									
Muutamien työkalupalkkien kuvakkeiden merkitys ei selviä kuvakkeen perusteella.		1								
Kaavion tallentamista kuvaksi kutsutaan Export-toiminnoksi.		1								
Lukittaessa piirrettävää näkymäelementtiä onnistunut lukitus ei näy ennen kuin osoitin siirretään pois valinnan kohdalta.					1					

Rational Rhapsodyn käytettävyydestä tehdyt havainnot										
Havainto	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Värejä ja hiiren osoitinta hyödynnetään palautteen antamiseen.					*					
Kaavion tallentaminen kuvana on vain pikavalikossa.	2									
Kaavion lisääminen etenee eri tavoin riippuen siitä valitaanko toiminto päävalikosta vai Browser-ikkunan pikavalikosta.				2						
Käyttöliittymä voi näyttää erilaiselta eri käyttökerroilla.				2						
Tarjolla ei ole peruutuspainiketta lisättäessä kaavioita Browser-ikkunan pikavalikon kautta.						2				
Piirrettäessä useita samoja elementtejä, on piirrettävä elementti valittava aina uudelleen.							2			
Ohje ei avaudu päävalikosta. PDF-tiedosto on saatavilla asennuskansiosta.										2
Yhteistoimintakaaviosta käytetään UML 1:n mukaista nimitystä.		1								
Välilehti suljetaan yläkulman rastipainikkeesta.				1						

Liite 4 Kaavioiden tallennuksessa tuetut tiedostomuodot

Taulukossa on ruksattu (x) tiedostomuodot, joihin kaavion tallentamisen on mahdollista. Kokeiltaessa toimimattomaksi havaittu tiedostomuoto on merkitty tähdellä (*).

Tiedostomuoto	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
BMP		x	x	x
EMF			x	x
EPS	x			
PS	x			
GIF	x	*		
HTML				
JPG		x	x	x
PDF				
PNG	x	x		
SVG	x			
TIFF				x
WMF			x	

Liite 5 Tuki koodin generoinnille

Taulukossa on esitetty kielet, joille mallinnusvälineellä voi generoida koodia perusasennuksessa (x) tai laajennosta käyttäen (*).

Ohjelmointikieli	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
C++	x	*	x	
C#	x	*	x	
Java	x	*	x	
PHP	x			

Liite 6 Elinkaarikustannuksiin vaikuttavat ominaisuudet

Mallinnustyökalujen käyttöjärjestelmätuki				
Käyttöjärjestelmä	**ArgoUML**	**Modelio**	**WhiteStarUML**	**Rational**
Windows XP tai uudempi	x	x	x	x
Linux	x	x		
Muut	x			

Graafissa on esitetty vertailtujen mallinnusvälineiden asennuspakettien koot megatavuina. ArgoUML vaatii lisäksi Java JRE:n, jos sitä ei ole valmiiksi asennettuna (*).