TAMPEREEN TEKNILLINEN YLIOPISTO

JANNE KOIVISTO

MICROSOFT EXCELIN LAAJENTAMINEN OHJELMISTOJEN MALLINNUSTYÖKALUKSI

Diplomityö

Tarkastajat: professori Kai Koskimies
yliassistentti Jari Peltonen

Tarkastajat ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston
kokouksessa 9. toukokuuta 2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KOIVISTO, JANNE: Microsoft Excelin laajentaminen ohjelmistojen mallinnustyökaluksi
Diplomityö, 61 sivua, 1 liitesivu
Toukokuu 2012
Pääaine: Hajautetut ohjelmistot
Tarkastajat: professori Kai Koskimies ja yliassistentti Jari Peltonen
Avainsanat: Mallinnustyökalu, Microsoft Excel, laajennus, hajautettu mallinnusympäristö

Ohjelmistotuotannossa käytetään perinteisten mallinnusohjelmien lisäksi tavallisia toimisto-ohjelmia. Taulukkolaskentaohjelmia käytetään, koska tietyn tyyppisiä tietoja on luontevinta esittää taulukkomuodossa. Laskentataulukoihin tehdyt kuvaukset voidaan nähdä osaksi ohjelmistosta laadittua mallia, vaikka niitä ei olekaan tehty varsinaisella mallinnustyökalulla. Koska mallinnustyökalut eivät ymmärrä näitä kuvauksia osaksi mallia, eri ohjelmissa olevia tietoja joudutaan muuntamaan ja ylläpitämään käsin. Muunnoksessa saatetaan menettää tietoa, koska käytetty mallinnustyökalu ei välttämättä tue taulukoissa käytettyjä kuvaustapoja. Muunnoksen jälkeen alkuperäiset taulukot jäävät irrallisiksi dokumenteiksi, eivätkä malliin tehdyt muutokset päivity niihin.

Taulukoita käytetään myös raporteissa, ja monet mallinnustyökalut tarjoavatkin mahdollisuuden taulukkomuotoisten raporttien generointiin. Generoidut raportit ovat kuitenkin staattisia, eivätkä muutokset päivity niihin. Lisäksi mallin tietoja ei voi muuttaa suoraan raporteista, vaan muutokset täytyy tehdä käytetyllä mallinnusohjelmalla.

Näiden ongelmien ratkaisemiseksi kehitettiin Microsoft Exceliin laajennus, joka toteutettiin osaksi Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella kehitettyä Trinity-työkaluympäristöä. Trinityyn on mahdollista integroida olemassa olevia sovelluksia, ja laajentaa niitä mallinnustoiminnoilla. Se on hajautettu ympäristö, joka sallii tietokannassa olevien tietojen yhtäaikaisen käsittelemisen useilla eri työkaluilla.

Exceliin toteutettu laajennus tarjoaa mahdollisuuden raporttien generointiin Trinityn malleista ja näkymistä sekä generoitavien elementtien suodattamisen erilaisilla kriteereillä. Lisäksi sen avulla on mahdollista muokata mallien tietoja suoraan generoiduissa raporteissa. Malleihin tehdyt muutokset päivitetään automaattisesti raportteihin. Lisäksi työkaluun toteutettiin mallinnustoimintoja, joiden avulla on mahdollista mallintaa Excelin avulla. Tämä diplomityö esittelee Excel-laajennuksen vaatimukset, sekä vaatimusten mukaisen toteutuksen toiminnallisuuden ja toteutuksen.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KOIVISTO, JANNE: Extending Microsoft Excel for a Software Modeling Tool
Master of Science Thesis, 61 pages, 1 Appendix page
May 2012
Major: Distributed Software
Examiner: Professor Kai Koskimies and assistant professor Jari Peltonen
Keywords: Modeling tool, Microsoft Excel, extension, distributed modeling environment

In software engineering, common office applications, such as spreadsheet applications, are used in context of software modeling, in addition to specific software modeling tools. Spreadsheets are used because tables are a natural form for presenting data. Descriptions created in spreadsheets can be seen as a part of a model; even though they are not made with an actual modeling tool. Data in separate tools must be converted and updated manually because modeling tools do not understand these descriptions. Information might be lost in the conversion because used modeling tool does not necessarily support description methods used in the spreadsheets. After the conversion is made the spreadsheets are not connected to the model and changes made to the model are not updated to them.

Tables are also used in reporting and some modeling tools offer a possibility to generate table-based reports from models. However, generated reports are static and changes made to the model are not updated to them. Additionally, the model data cannot be changed by using the reports directly, but the changes must be done with the modeling tool.

A Microsoft Excel extension was developed to solve these problems. It was integrated to the Trinity tool environment developed at the Department of Software Systems at Tampere University of Technology. Trinity offers a possibility to integrate existing applications and extend them with modeling functionality. Trinity is a distributed environment that allows manipulation of model data concurrently with several tools.

Implemented Excel extension allows users to generate reports from models and views in the Trinity environment. In the reports, the generated elements can be filtered with different kind of criteria. Additionally, the extension allows users to edit model data directly from generated reports, and the changes made to models in other tools are propagated automatically to the generated reports. Also, some modeling features were implemented to the extension to allow modeling by using spreadsheets of Microsoft Excel. This master thesis presents requirements, as well as implemented functionality conforming the requirements and implementation of the Microsoft Excel extension.

ALKUSANAT

Haluan kiittää kaikkia Trinity-projektissa mukana olleita henkilöitä. Kiitokset myös työn tarkastajille Kai Koskimiehelle ja Jari Peltoselle työstä annetuista kommenteista ja parannusehdotuksista. Lisäksi haluan kiittää kaikkia muita työn kielioppiin ja sisältöön vaikuttaneita henkilöitä. Lopuksi haluan kiittää perhettäni ja ystäviäni, jotka jaksoivat muistuttaa minua diplomityöstäni ja kannustaa sen tekemisessä.

Tampereella 17. toukokuuta 2012

_____

Janne Koivisto

SISÄLLYS

TERMIT JA LYHENTEET

Avainsana (Tag)	Elementteihin voidaan liittää erilaisia avainsanoja, joiden perusteella niitä voidaan ryhmitellä ja etsiä. Avainsanalle voi olla asetettu tyyppi (TagType) tai se voi olla tyypitön.
Baseline-kopiointi	Trinityn tarjoama mallien kopiointitapa. Alkuperäisestä mallista tehdään kopio, jota käytetään vertailukohtana kopioidulle mallille. Kopioituun malliin tehtyjä muutoksia voidaan myöhemmin verrata alkuperäiseen baseline-malliin.
CASE-työkalu	Computer-Aided Software Engineering. CASE-työkalut ovat työkaluja, joita käytetään ohjelmistokehityksen apuna.
Laskentataulukko (Worksheet)	Excel työkirjan välilehdellä sijaitseva soluista muodostuva taulukko.
Mallielementti	Mallit rakentuvat mallielementeistä, joilla on jokin tyyppi, esimerkiksi UML-luokka. Mallielementin tyyppi määrittää mallielementin ominaisuudet. Mallissa mallielementtien ominaisuuksille on asetettu arvoja sekä niiden välille on luotu erilaisia suhteita.
MOF	MetaObject Facility. MOF on UML:n käyttämä metametamalli, jonka ilmentymä UML:n mallinnuskieli on.
Näkymäelementti	Näkymät koostuvat näkymäelementeistä, joihin talletetaan mallielementtien visuaaliseen esitykseen liittyvät tiedot. Näkymäelementtejä käytetään esittämään mallielementtien tietoja näkymissä. Trinityssä mallielementille luodaan ensisijainen näkymäelementti sen luomisen yhteydessä. Muut samaan mallielementtiin viittaavat näkymäelementit ovat viittaavia näkymäelementtejä.
Excelin Oliomalli (Excel Object Model)	Excelin oliomalli tarjoaa oliopohjaisen pääsyn käynnissä olevan Excel-prosessin eri osiin ja mahdollistaa niiden hallitsemisen toisesta ohjelmasta.
OMG	Object Management Group. OMG on avoin kansainvälinen konsortio, joka keskittyy malliperustaisten standardien määrittelyyn. Se tarjoaa mallintamiseen liittyviä spesifikaatioita, joista UML on suosituin.

ORM	Oliorelaatiomuunnos (Object-relational mapping). Oliorelaatiomuunnosta käytetään muuttamaan relaatiotietokannassa oleva tieto oliokieltä käyttävän ohjelmoijan kannalta helppokäyttöisempään oliomuotoon.
Solu (Cell)	Laskentataulukon yksittäinen solu, joka yksilöidään rivin numerolla ja sarakkeen kirjaimella (esimerkiksi A1). Solu voi sisältää erityyppisiä arvoja kuten teksti, päivämäärä ja desimaaliluku. Sekä itse solu että sen arvo sisältävät lukuisia niiden muotoiluun liittyviä ominaisuuksia.
Solualue (Range)	Yhdestä tai useammasta solusta muodostava alue, johon voidaan viitata laskentataulukossa. Excelin oliomallissa soluja käsitellään solualeina.
Template-elementti	Trinityssä template-elementti määrittää sitä käyttävien näkymäelementtien oletusulkoasun. Lisäksi se määrittää näkymäelementtiin liittyvän mallielementtien tyypin.
Trinity	Trinity on Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella kehitetty hajautettu mallinnus- ja työkaluympäristö.
Työkirja (Workbook)	Excel-dokumentti, joka koostuu välilehdillä sijaitsevista laskentataulukoista ja kaavioista.
UML	Unified Modeling Language. UML on OMG:n määrittelemä, ohjelmistojen mallintamiseen yleisesti käytetty mallinnuskieli.

1. JOHDANTO

Ohjelmistotuotannossa käytetään taulukoita tietojen esittämiseen ja käsittelyyn, koska ne ovat luontevin esitystapa tietyn tyyppisten tietojen esittämiseksi. Taulukoita voidaan ajatella graafisen ja tekstimuotoisen esitystavan välimuotona – niillä on määrätty rakenne, mihin on yhdistetty tekstiä. Taulukoidusta tiedosta saa selville nopealla vilkaisulla esimerkiksi tietojen välisiä suhteita jo pelkän taulukon rakenteen perusteella. Yleisiä käyttökohteita taulukoille ovat muun muassa tietohakemistot mallien tiedoista sekä esimerkiksi ohjelmiston ominaisuuksien, poikkeusten, parametrien ja vaatimusten listaaminen.

Kun vapaamuotoisesta taulukoidusta tiedosta luodaan ohjelmistoprosessin myöhemmässä vaiheessa formaalimpi malli, joudutaan tiedot yleensä syöttämään uudelleen erilliseen mallinnustyökaluun. Käytetty mallinnustyökalu ei välttämättä tue kaikkia taulukossa käytettyjä epäformaaleja kuvaustapoja, jolloin muunnoksessa menetetään tietoa. Mallin luomisen jälkeen alkuperäiset taulukot jäävät irrallisiksi dokumenteiksi. Taulukko täytyy päivittää manuaalisesti, mikäli sen halutaan pysyvän ajan tasalla muunnoksen jälkeen. Pahimmassa tapauksessa mallissa ja taulukossa olevat tiedot ovat ristiriitaisia, eikä voida olla enää varmoja mitkä tiedoista pitävät paikkansa.

Perinteiset mallinnustyökalut tarjoavat usein mahdollisuuden raporttien generointiin mallin tiedoista. Niiden ongelmana on kuitenkin, että generoidut raportit ovat staattisia eikä malliin tehdyt muutokset päivity niihin. Lisäksi mallin tietoja ei voida muuttaa suoraan raportista, vaan muutokset pitää tehdä erillisen mallinnustyökalun avulla.

Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella on tutkittu mallintamisen työkalutukea jo usean vuoden ajan ja siellä kehitetään parhaillaan hajautettua Trinity-työkaluympäristöä. Trinity ei ole itsessään mallinnustyökalu tai muukaan CASE-työkalu, vaan se tarjoaa mahdollisuuden erilaisten työkalujen integroimiseksi osaksi ympäristöä. Ympäristö voidaan räätälöidä organisaation tarpeiden mukaan, ja työtä pystytään tekemään eri vaiheisiin parhaiten soveltuvilla työkaluilla perinteisen työkalulähtöisen tavan sijaan, jossa työkalut pakottavat käyttäjät työskentelemään tietyllä tapaa.

Trinity-ympäristöön haluttiin integroida työkalu automaattisesti päivittyvien taulukkomuotoisten raporttien generointiin ympäristön malleista. Lisäksi haluttiin antaa käyttäjille mahdollisuus käsitellä mallien tietoja taulukkomuodossa. Integroitavaksi työkaluksi valittiin Microsoft Excel, jonka laajentamista raportointi- ja mallinnustyökaluksi tämä työ käsittelee. Excel-laajennukseen toteutettiin toiminnot raporttien generoinnille ja generoitavien elementtien suodattamiselle. Lisäksi mallien tietoja on mahdollista

muokata Excel-näkymien kautta yhtä aikaa ympäristön muiden käyttäjien kanssa. Trinityn Excel-laajennusta on tarkoitus käyttää yhdessä Trinity ympäristön muiden työkalujen kanssa ja käyttäjät voivat valita tilanteeseen kulloinkin parhaiten soveltuvan työkalun.

Luvussa kaksi esitellään tärkeimmät taustatiedot ja teoria työn ymmärtämisen kannalta. Luvussa kolme pohditaan perinteisten mallinnustyökalujen ja -ympäristöjen ongelmia globaalien organisaatioiden kannalta sekä esitellään kyseisten ongelmien ratkaisemiseksi kehitetty Trinity-ympäristö. Luku neljä käsittelee työn ongelmaa sekä kuvaa työkalulle asetetut vaatimukset. Luvussa viisi kuvataan toteutetun työkalun tietosisältö sekä käyttö ja tärkeimmät toiminnot. Luvussa kuusi kuvataan Trinityyn integroitaville työkaluille toteutetun ohjelmistokehyksen arkkitehtuuri sekä kerrotaan, kuinka kehys erikoistettiin Trinityn Excel-laajennukseksi. Luvussa seitsemän arvioidaan Excel-laajennuksen toteutusta sekä pohditaan sen tulevaisuuden näkymiä. Luku kahdeksan sisältää työn yhteenvedon.

2. TAUSTAKÄSITTEET JA -TEKNIIKAT

Tässä luvussa esitellään lyhyesti tämän työn kannalta tärkeimmät taustakäsitteet ja tekniikat. Ensin käydään lyhyesti läpi oleelliset mallinnuksen peruskäsitteet ja kerrotaan mallien käytöstä ohjelmistotuotannossa. Sen jälkeen esitellään UML-mallinnuskieli. Lisäksi esitellään Microsoft Excelin ominaisuudet sekä sen oliopohjainen laajennusrajapinta – Excel Object Model. Lopuksi esitellään työssä käytetyt arkkitehtuuri- ja suunnittelumallit sekä kerrotaan ohjelmistokehyksistä.

2.1 Ohjelmistojen mallintaminen

Ohjelmistotuotannossa käytetään paljon erilaisia *malleja* (model) kuvaamaan järjestelmiä ja niiden osia. Mallit ovat yleensä reaalimaailman yksinkertaistuksia, joista on abstrahoitu pois kaikki mallinnettavan kohteen kannalta epäolennainen. Ne rajoittuvat yleensä johonkin tiettyyn näkökulmaan, joka kuvaa järjestelmän kannalta merkityksellistä ominaisuutta. Malleilla voidaan kuvata järjestelmien rakennetta tai toiminnallisuutta, ja ne voivat käsitellä järjestelmän dynaamisia tai staattisia ominaisuuksia. [1]

Näkymä (view) on mallin esitys, joka rajoittuu usein vain tiettyyn mallin osaan. *Näkymätyyppi* (viewtype) määrää tietyn esitysmuodon kuvaukselle, jota näkymä seuraa. [1] Yhteen malliin voi liittyä useita näkymiä, joiden kautta mallin tietoja voidaan käsitellä. Ne voivat olla erityyppisiä ja tarkoitettu käytettäväksi eri mallinnustyökaluilla. Esimerkiksi luokkamalli voi sisältää luokkakaavionäkymiä ja tietohakemistonäkymiä. Kaavionäkymissä elementit esitetään graafisesti ja tietohakemistonäkymissä esimerkiksitaulukkomuotoisena. Saman mallin eri näkymät voivat olla korkean tason yleiskuvauksia tai tarkennettuja suunnittelutason kuvauksia, missä samoista elementeistä esitetään erilaisia tietoja.

Ohjelmistojen mallintaminen on yleensä iteratiivinen prosessi [2], jossa lähdetään liikkeelle esimerkiksi käsitteellisestä mallintamisesta. Tällöin määritellään mallinnettavan järjestelmän tärkeimmät käsitteet sekä niiden väliset suhteet. Myöhemmässä vaiheessa käsitteellistä mallia voidaan käyttää pohjana esimerkiksi tietokantaratkaisujen suunnittelussa, jolloin käsitteet tarkennetaan vastaamaan relaatiotietokannan tauluja. Käsitteellisestä mallista voidaan myös edetä tarkempaan suunnittelutason malliin, jolloin käsitteitä tarkennetaan lisäämällä niihin ominaisuuksia ja toimintoja sekä uusia suunnittelutason vaatimia käsitteitä.

2.1.1 Mallien käyttö ohjelmistotuotannossa

Booch et al. [2] mukaan tärkein syy ohjelmistojen mallintamiseen on kehitettävän järjestelmän ymmärtäminen. Lisäksi he esittävät neljä mallien tekemisestä saatavaa hyötyä:

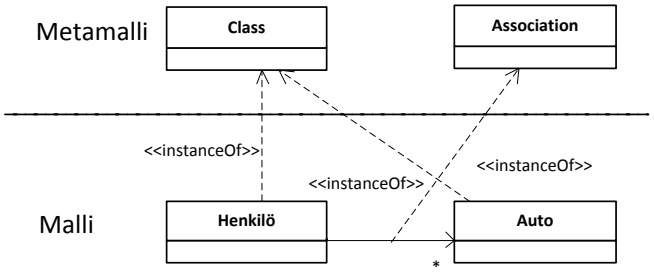
1. Mallit helpottavat visualisoimaan järjestelmiä sellaisina kuin ne ovat tai sellaisina kuin niiden halutaan olevan.
2. Mallit mahdollistavat järjestelmän rakenteen tai toiminnan määrittelyn.
3. Mallit antavat muotin, joka ohjaa järjestelmän toteuttamista.
4. Mallit dokumentoivat tehdyt päätökset.

Koskimies et. al. [1] mukaan malleilla on keskeinen merkitys kommunikaation mahdollistavina ohjelmistoartefakteina. Mallit kuvaavat ongelma-alueet ja niihin liittyvät ratkaisut sekä antavat käsitteistön ja sanaston, joiden avulla järjestelmästä voidaan puhua. Mallien käyttötapa ja abstraktiotaso voi vaihdella aina korkeimman tason arkkitehtuurikuvauksista matalimman tason yksityiskohtaisiin suunnitelmiin. Malleja voidaan käyttää kehitettävän järjestelmän luonnostelun, määrittelyn ja dokumentoinnin lisäksi myös ohjelmakoodin generoimiseen, kunhan mallit on tehty riittävällä tarkkuudella.

2.1.2 Mallintamisen metatasot

Mallin ja siihen liittyvän *mallinnuskielen* välistä suhdetta voidaan kuvata hierarkiana, jossa malli on hierarkiatasolla sen yläpuolella sijaitsevan mallinnuskielen ilmentymä. Mallinnuskieli on täten yhtä metatasoa korkeammalla kuin siihen perustuva malli, ja siksi sitä voidaan kutsua myös *metamalliksi*. Metamalli voi myös olla korkeamman tason metamallin ilmentymä. Tätä seuraavan metatason mallia voitaisiin kutsua *metametamalliksi*. Mallihierarkia voisi jatkua tähän tapaan rajattomasti. Käytännön sovelluksissa rajoitutaan kuitenkin yleensä metametamallin tasolle, joka pystyy määrittelemään itsensä. [3,4]

Malli sisältää tyypillisesti mallielementtejä, jotka tehdään luomalla ilmentymiä metamallin elementeistä. Metamallin elementit ovat vastaavalla tavalla ilmentymiä metametamallin elementeistä. [4] Kuva 2.1 esittää esimerkin mallin ja sen metamallin välisestä suhteesta. Mallissa sijaitsevat mallielementit Henkilö ja Auto ovat metamallin määrittämän Class-mallielementin ilmentymiä. Niiden välillä oleva suhde on puolestaan metamallissa sijaitsevan Association-mallielementin ilmentymä.

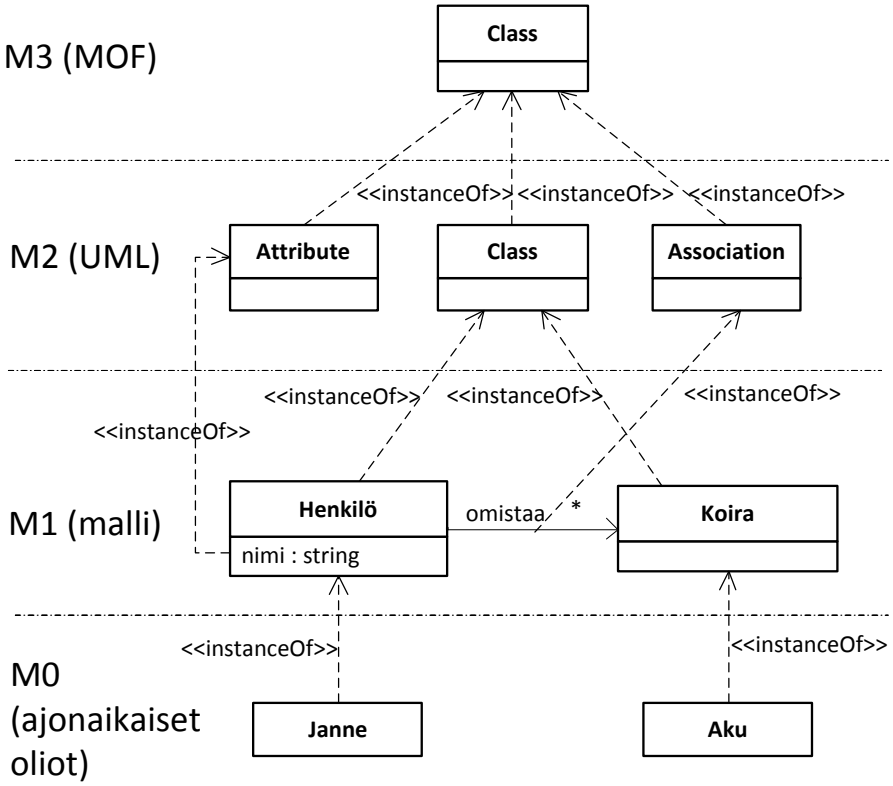


Kuva 2.1. Mallin ja meta-mallin välinen suhde. Mukailtu lähteestä [4].

2.1.3 Unified Modeling Language

Unified Modeling Language (UML) [5] on Object Management Groupin vuonna 1997 standardoima graafinen mallinnuskieli. Se on tunnetuin ja eniten käytetty mallinnuskieli ohjelmistoteollisuudessa [2]. UML:n versio 2.3 koostuu yhteensä 14:sta erilaisesta kaaviotyypistä, jotka voidaan jakaa ohjelmistojen rakennetta ja käyttäytymistä kuvaaviin [6]. Kaikki kaaviotyypit kuuluvat samaan mallinnuskieleen, ja tämän vuoksi kaaviotyyppien elementtejä voidaan käyttää kaavioissa sekaisin, kunhan ne eivät riko UML:n metamallin sääntöjä.

UML on metamallipohjainen mallinnuskieli. Sen metamalli perustuu UML:n metametamalliin, joka on nimeltään MetaObject Facility (MOF) [**7**]. UML:n metatasot kuvataan nelitasoisena hierarkiana. Kuva 2.2 esittää kyseisen hierarkian ja esimerkkielementit sen eri tasoilla. Ylimmällä M3-tasolla sijaitsee MOF. Sen alapuolella olevan M2-tason, eli UML:n metamallin, mallielementit ovat MOF:in mallielementtien ilmentymiä. M1-sasolla sijaitsee malli, jonka mallielementit ovat vastaavasti M2-tason mallielementtien ilmentymiä. Alimmalla M0-tasolla sijaitsee mallissa määriteltyjen mallielementtien ajonaikaiset ilmentymät. [4]



Kuva 2.2. UML:n eri metatasot. Mukailtu lähteestä [4].

2.2 Microsoft Excel ja Excelin oliomalli

Microsoft Excel, jatkossa Excel, on Microsoft Office -toimisto-ohjelmapaketin mukana tuleva taulukkolaskentaohjelma. Se on tarkoitettu taulukkomuotoisen tiedon esittämiseen ja käsittelyyn. Excel-dokumentti eli *työkirja* (workbook) koostuu välilehdistä, jotka

sisältävät *laskentataulukoita* (worksheet) tai kuvaajia (chart). Laskentalukoissa tiedot esitetään *soluissa* (cell), jotka on järjestetty numeroin merkityille riveille ja kirjaimin merkityille sarakkeille.

Excel tarjoaa joukon funktioita tietojen käsittelemiseen sekä toimintoja solujen, tietojen ja laskentataulukoiden muotoilemiseen. Lisäksi se mahdollistaa erilaisten kuvaajien tekemisen laskentataulukoiden tiedoista. [8]

Exceliä käytetään erittäin paljon yritysmaailmassa. PCWorldissa julkaistun artikkelin mukaan Microsoft Office on käytetyin toimisto-ohjelmisto yritysmaailmassa. Forrester Researchin tekemän tutkimuksen mukaan kesäkuussa 2009 80%:ssa yrityksistä oli käytössä jokin versio Officesta, ja noin 80%:ssa näistä oli käytössä Officen uusin versio. [9]

Microsoft Excel Object Model [10] eli oliomalli tarjoaa oliopohjaisen ohjelmointirajapinnan Exceliin .NET-ympäristössä [11]. Sitä on mahdollista käyttää .NET:in tukemilla ohjelmointikielillä.

Oliomallin tärkeimpiä luokkia ovat *Application*, *Workbook*, *Worksheet* ja *Range* [12]. Excelin perustoiminnot keskittyvät näiden luokkien jäseniin (member) eli metodeihin ja *luokan ominaisuuksiin* (property) [13]. Luokan ominaisuus on C#-kielen [14] mekanismi, joka mahdollistaa get- ja set-metodien toteuttamisen helposti luokkien jäsenmuuttujille. Edellä mainittujen luokkien lisäksi oliomalli tarjoaa satoja muita luokkia [15], joiden avulla kehittäjät pääsevät käsiksi Excelin eri osiin ja toimintoihin.

Application-olio toimii kääreenä käynnissä olevalle Excel-sovellukselle. Se tarjoaa pääsyn käynnissä olevan Excel-sovelluksen tietoihin ja asetuksiin. Application luokan jäsenet voidaan jakaa Excelin tilaa ja näkymää ohjaaviin, olioita palauttaviin, toimintoja suorittaviin, tiedostojen käsittelyyn liittyviin sekä muihin jäseniin. Tärkeimpiä olioita palauttavia jäseniä ovat muun muassa Selection-, Sheets- ja Workbooks-ominaisuudet, joiden avulla voi palauttaa Excel-sovelluksen eri osia. [12]

Jäsenien lisäksi Application-luokka tarjoaa joukon tapahtumia (event). Laskentataulukon käyttäytymiseen liittyviä tapahtumia ovat muun muassa: SheetActivate, SheetChange ja SheetSelectionChange. SheetActivate-tapahtuma laukaistaan, kun laskentataulukko aktivoidaan. SheetChange-tapahtuma laukaistaan, kun solualueen arvo muuttuu laskentataulukossa. Sen mukana annetaan muuttuneeseen alueeseen viittaava Range-tyyppinen olio sekä laskentataulukko, johon muutos kohdistui. SheetSelectionChange-tapahtuma puolestaan laukaistaan, kun valittu alue muuttuu laskentataulukossa. Myös Workbook- ja Worksheet-luokka tarjoavat nämä samat laskentataulukon käyttäytymiseen liittyvät tapahtumat.

Workbook-olio edustaa yksittäistä työkirjaa Excelissä. Workbook-luokka tarjoaa noin 90 ominaisuutta. Näistä suurin osa liittyy spesifisiin tapauksiin, joita suurin osa kehittäjistä ei tule edes miettineeksi [12]. Workbook-luokan tärkeimpiä ominaisuuksia ovat dokumentin asetuksiin kuten sen nimeen ja tiedostonimeen liittyvät ominaisuudet, solutyylien käsittelyyn liittyvät ominaisuudet sekä Sheets-ominaisuus, jonka avulla pääsee käsiksi työkirjan laskentataulukoita sisältäviin välilehtiin.

Worksheet-olio edustaa yksittäistä työkirjan laskentataulukkoa. Worksheet-luokka sisältää suuren määrän samoja tai samankaltaisia jäseniä, jotka myös Application- ja Workbook-luokat tarjoavat. Se tarjoaa useita luokan ominaisuuksia, jotka palauttavat toisia olioita. Näitä ovat esimerkiksi Comments-ominaisuus, jolla voi hakea laskentataulukon soluihin liitettyjä kommentteja sekä Outline-ominaisuus, jolla voi näyttää tai piilottaa eri tavoin ryhmiteltyjä solualueita.

Range-luokka on käytetyin oliomallin luokista Excel-laajennusten kehittämisessä [12]. Range-olion avulla on mahdollista käsitellä laskentataulukoiden solualueita. Se voi edustaa yksittäistä solua, riviä, saraketta tai joukkoa soluja, jotka voivat sijaita vierekkäisissä soluissa tai erillään ja jopa eri laskentataulukoissa. Kuten aiemmin kuvatut luokat, myös Range-luokka sisältää huomattavan suuren määrän jäseniä.

Ennen kuin oliomallin kautta voidaan käsitellä mitään laskentataulukon aluetta, täytyy alueeseen viittaavaan Range-olioon päästä käsiksi. Range-olion hakemiseen on useita tapoja. Kutsumalla Application-olion ActiveCell-metodia saadaan laskentataulukossa valittuna olevaan alueeseen viittaava Range-olio. Worksheet-olion get_Range-metodilla voidaan hakea laskentataulukossa oleviin soluihin viittaava Range-olio antamalla parametrina solun osoite A1-muodossa tai alueen osoite muodossa A1:B12. Worksheet-olion Cells-ominaisuudella voidaan hakea tietty solu laskentataulukosta indeksoimalla sitä muodossa Cells[rivinumero, sarakenumero]. Range-oliosta voidaan hakea samaan tapaan toinen alue indeksoimalla Cells, Rows tai Columns -ominaisuuksia. Näiden lisäksi on mahdollista hakea muun muassa alueiden unioneita ja leikkauksia Application-luokan Union- ja Intersect-metodeilla, joille annetaan tutkittavat Range-oliot parametreina.

Kun kehittäjä on saanut haluamaansa alueeseen viittaavan Range-olion haltuunsa, sille on mahdollista tehdä esimerkiksi seuraavia toimintoja. Alueen muotoilua voi käsitellä niille tarkoitettujen luokan ominaisuuksien avulla. Solujen arvon voi hakea ja sitä voi muuttaa Value2-ominaisuuden avulla. Range-luokka tarjoaa myös vanhemman, parametrisoidun Value-ominaisuuden solun arvon hakemiseen. Solun arvon muotoilua voidaan muuttaa hakemalla Font-olio Rangen Font-ominaisuudella, ja vaihtamalla sen ominaisuuksien arvoja.

Visual Studio Tools for Office [16] laajentaa monia oliomallin tarjoamia luokkia isäntäesineiksi (host item) ja isäntäohjaimiksi (host control) [17]. Laajennetut luokat tarjoavat lisää toiminnallisuutta ja tapahtumia (event). Esimerkiksi Range-luokasta tarjotaan laajennettu NamedRange-luokka. Se tarjoaa Selected-, Deselected- ja SelectionChange-tapahtumat. Näistä ensimmäinen laukaistaan valinnan tullessa olion alueelle, seuraava valinnan poistuessa siitä ja viimeinen valinnan vaihtuessa olion solualueen sisällä.

2.3 Ohjelmistoarkkitehtuureihin liittyvät tekniikat

Suunnittelumallit kuvaavat yleiskäyttöisiä ja hyviksi todettuja ratkaisuja tietyissä konteksteissa usein toistuviin ongelmiin. Ne on pyritty nimeämään selkeästi, jotta niiden

avulla voitaisiin dokumentoida helposti monimutkaisia ratkaisuja ja suunnittelijat voisivat keskustella järjestelmistä korkeammalla abstraktiotasolla käyttäen suunnittelumallien nimiä. *Arkkitehtuurimallit* puolestaan kuvaavat korkeimman tason suunnitteluratkaisuja, jotka määrittelevät sovellusten rakenteen.

Suunnittelumallit yleistyivät ohjelmistoteollisuudessa niin sanotun neljän koplan (Gang of Four) julkaistua ensimmäisen kirjan ohjelmistoissa käytettävistä suunnittelumalleista vuonna 1994. Tämä suunnittelumallien perusteoksena pidettävä Design Patterns-kirja luokittelee suunnittelumallit niiden tarkoituksen perusteella luonti- rakenne- ja käyttäytymismalleihin. Kirja kuvaa yhteensä 23 erilaista suunnittelumallia, joista kuvataan mallin nimi, ratkaistava ongelma, yleiskäyttöinen ratkaisu kyseiseen ongelmaan sekä mallin käytöstä aihetuvat seuraukset.

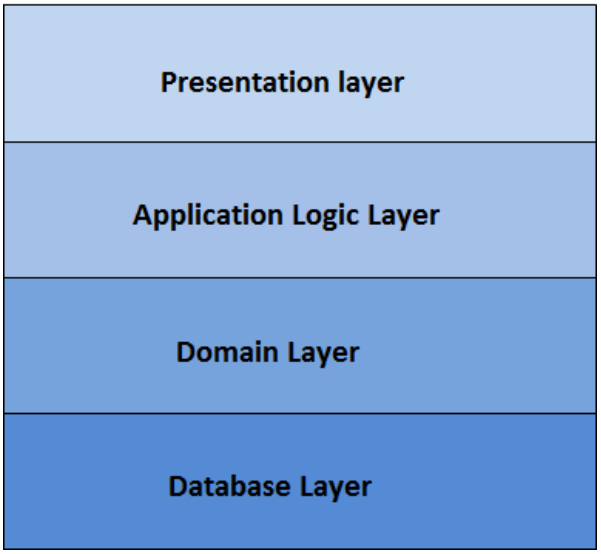
Tämän työn suunnittelussa ja toteutuksessa käytettiin useita suunnittelu- ja arkkitehtuurimalleja, joista olennaisimmat kuvataan seuraavissa aliluvuissa. Lisäksi selitetään ohjelmistokehyksen käsite työssä kuvatun ratkaisun ymmärtämiseksi.

2.3.1 Kerrosarkkitehtuuri

Kerrosarkkitehtuuri (Layers) helpottaa sovelluksen komponenttien ryhmittelyä. Sovelluksen alijärjestelmät voidaan jakaa eri abstraktiotasoille siten, että ylemmällä abstraktiotasolla olevat *kerrokset* käyttävät hyväkseen alempien kerrosten tarjoamia matalamman abstraktiotason palveluja. Täten sovelluksen arkkitehtuuri koostuu erillisistä kerroksista, joissa abstraktiotaso nousee kerroshierarkiassa ylöspäin noustessa. [18]

Koskimies & Mikkosen [19] mukaan kerrosarkkitehtuurien perusideasta joudutaan yleensä aina poikkeamaan. Poikkeamia on kahdenlaisia: alemmasta kerroksesta ylempään kulkevat palvelukutsut (hierarchy breach) sekä kerroksen ohittavat palvelukutsut (bridging). Heidän mukaansa kerrosten ohittaminen on usein tarpeellista tehokkuussyistä, sillä palvelu löytyy usein tehokkaampana alemmilta kerroksilta. Kerrosten ohittamista maltillisesti käytettynä ei yleensä pidetä vakavana kerrosarkkitehtuurista poikkeamisena. Alemmasta kerroksesta ylempään kulkevat kutsut ovat sen sijaan vakava kerrosarkkitehtuurin ongelma, jos alempi kerros tulee riippuvaiseksi ylemmästä.

Buschmann et. al. [18] mukaan liiketoiminnan informaatiojärjestelmissä käytetään usein kerrosarkkitehtuuria, joissa alimmalla tasolla sijaitsee yrityksen tietokanta. Monet erilaiset sovellukset käyttävät suoraan tätä samaa tietokantaa esimerkiksi Asiakas-palvelin-mallin mukaisesti, jolloin arkkitehtuuria kutsutaan kaksikerrosarkkitehtuuriksi. Yleensä sovelluskerroksen ja tietokantakerroksen välille lisätään kolmas kerros helpottamaan tietokannan käyttöä eri sovelluksissa. Tämä kerros mallintaa kohdealueen käsitteellistä rakennetta ja sen tehtävänä on vähentää suoria riippuvuuksia sovellusten ja tietokannan välillä. Myös ylin kerros jaetaan usein kahteen osaan: käyttöliittymän sisältävään esitystapakerrokseen ja sovelluksen liiketoimintalogiikan sisältävään sovelluslogiikkakerrokseen. Kuva 2.3 näyttää esimerkin tällaisesta nelikerrosarkkitehtuurista.



Kuva 2.3. Esimerkki liiketoiminnan informaatiojärjestelmän nelikerrosarkkitehtuurista. Mukailtu lähteestä [18].

Koskimies & Mikkonen [19] toteavat, että kyseisen kaltainen ratkaisu mahdollistaa uusien sovellusten luomisen samalle sovellusalueelle suoraan kahden alimman kerroksen päälle. Lisäksi esimerkiksi tietokantakerroksen käyttämä tiedon tallennustapa voidaan vaihtaa tiedostopohjaisesta ratkaisusta relaatiotietokannaksi ilman, että siitä on vaikutusta arkkitehtuurin muihin kerroksiin.

Buschmann et. al. [18] mainitsee kerrosarkkitehtuurin käyttämisestä saataviksi hyödyiksi muun muassa seuraavat asiat:

- Kerroksia voidaan käyttää uudelleen. Varsinkin alempien kerrosten matalamman abstraktiotason palveluita voidaan käyttää hyödyksi toisissa sovelluksissa.
- Riippuvuudet saadaan pidettyä paikallisina, koska alemmat kerrokset eivät riipu ylemmistä ja kerrosten väliset riippuvuudet voidaan hoitaa standardirajapinnoilla.
- Yksittäinen kerros voidaan vaihtaa toiseen samat palvelut tarjoavaan kerrokseen. Mikäli kerrosten rajapinnat eivät ole suoraan yhteensopivia, apuna voidaan käyttää aliluvussa 2.3.4 esiteltävää Sovitin-suunnittelumallia.

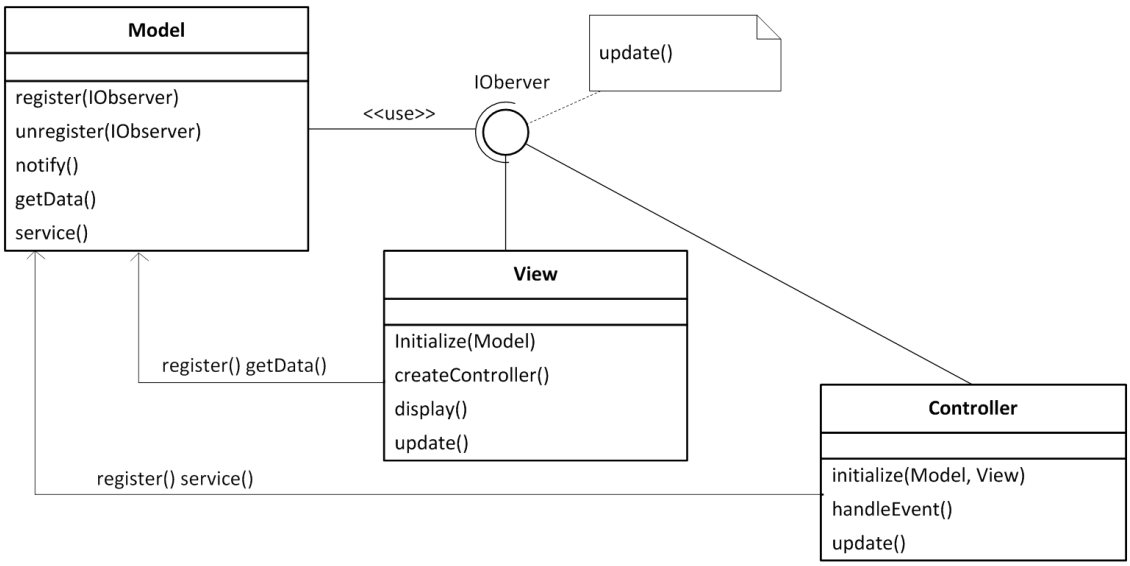
Kerrosarkkitehtuurin käytöstä voi kuitenkin seurata myös ongelmia. Kerrosarkkitehtuuri ei ole yhtä suorituskykyinen kuin monoliittinen rakenne, koska palvelukutsut ja niihin tulevat vastaukset saattavat joutua kulkemaan monen kerroksen läpi. Alemmilla kerroksilla saatetaan myös suorittaa operaatioita, joita ylemmät kerrokset eivät tarvitse, ja samoja operaatioita voidaan joissain tapauksissa suorittaa turhaan monella eri kerroksella. Toinen ongelma on, että alemman kerroksen toiminnan muuttumisesta voi aiheutua muutostarpeita kaikkiin muuttuneen kerroksen yläpuolella sijaitseviin kerroksiin. Tämä ei kuitenkaan ole suuri ongelma, koska ylemmät kerrokset voidaan yleensä suojata alempien kerrosten muutoksilta. [18]

2.3.2 Malli-näkymä-ohjain-arkkitehtuuri

Malli-näkymä-ohjain-arkkitehtuuri (Model-View-Controller, MVC) jakaa interaktiivisen sovelluksen kolmeen eri osaan: *malliin* (model), *näkymään* (view) ja *ohjaimeen* (controller). Malli sisältää sovelluksen tiedot ja niiden käsittelyyn liittyvät toiminnot. Näkymä toimii käyttöliittymänä. Ohjaimen tehtävänä taas on käsitellä käyttäjiltä tulevat syötteet sekä pitää mallin ja näkymän tilat synkronoituna toistensa kanssa [18].

Malli-näkymä-ohjain-arkkitehtuurista on olemassa useita erilaisia variaatioita. Yksi tapa on käyttää sitä yhdessä T*arkkailija-suunnittelumallin* (Observer) [20] kanssa. Tarkkailija toteutetaan rajapintana, joka rekisteröidään kuuntelemaan kohteen muutoksia. Kohde ilmoittaa muutoksista tarkkailija-rajapinnan kautta. Microsoft .NET sisältää Tarkkailija-suunnittelumallin hyödyntämiseen helppokäyttöisemmän toteutustavan tapahtumien (event) ja delegaattien (delegate) avulla. Erillistä tarkkailija-rajapintaa ei tarvitse toteuttaa, vaan riittää että määritellään tapahtumat ja delegaatit. Delegaatti on funktio-osoitin, jonka tarkkailija sitoo tiettyyn kuunneltavan kohteen tapahtumaan, ja jota kohde kutsuu tapahtuman realisoituessa. Delegaatin tyyppi määritellään kohteen tapahtuman määrittelyssä. Tarkkailija toteuttaa sitä vastaavan funktion, joka suoritetaan tapahtuman lauetessa. [21]

Kuva 2.4 näyttää esimerkin Malli-näkymä-ohjain-arkkitehtuurista, joka käyttää Tarkkailija-suunnittelumallia. Näkymä (View) ja ohjain (Controller) toteuttavat tarkkailija-rajapinnan (IObserver). Ne rekisteröityvät mallille (Model) kuuntelemaan muutoksia. Mallin tietojen muuttuessa se pyytää näkymää ja ohjainta päivittämään itsensä, tarkkailija-rajapinnan update()-funktion avulla.



Kuva 2.4. Esimerkki tarkkailija-suunnittelumallia käyttävästä Malli-näkymä-ohjain-arkkitehtuurista. Mukailtu lähteestä [19].

Buschmann et. al. [18] mainitsee Malli-näkymä-ohjain-arkkitehtuurin hyödyiksi muun muassa seuraavat asiat:

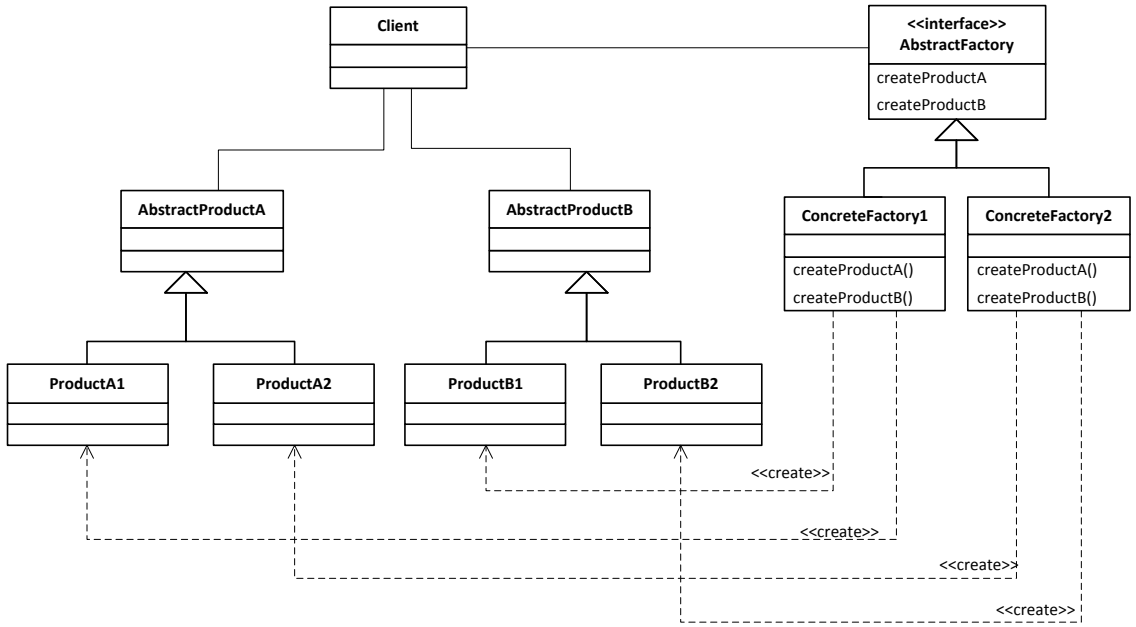
- Yhdellä mallilla voi olla useita näkymiä, joita voidaan luoda myös ajonaikana dynaamisesti.
- Kaikki malliin liittyvät näkyvät on synkronoitu, koska malli tiedottaa muutoksista kaikille niistä kiinnostuneille.
- Näkymä ja ohjain voidaan vaihtaa esimerkiksi toiselle graafiselle alustalle soveltuvaksi, koska malli ei ole niistä riippuvainen.
- Se soveltuu hyvin sovelluskehysten perustaksi. Päätoiminnallisuus voi sijaita kehyksessä mallina. Näkymä ja ohjain voidaan puolestaan toteuttaa sovelluskohtaisina.

Malli-näkymä-ohjain-arkkitehtuurin ongelmana on, että mallin käyttö saattaa lisätä tarpeetonta monimutkaisuutta sitä käyttäviin järjestelmiin. Lisäksi kaikkia näkymiä ei välttämättä tarvitsisi päivittää muutoksissa, ja nämä turhat päivitykset, esimerkiksi näkymän ollessa pienennettynä, heikentävät suorituskykyä. Näkymät ja ohjaimet ovat läheisesti toisistaan riippuvia, eikä niiden uudelleenkäyttö yleensä onnistu yksittäin. Ne ovat myös riippuvaisia mallista, koska ne käyttävät sitä suoraan. [18]

2.3.3 Tehdas-suunnittelumallit

Erilaisia tehdas-suunnittelumalleja (factory pattern) käytetään luontiriippuvuuksien purkamiseksi luokkien väliltä. Niiden periaatteena on tarjota rajapinta luokkien luomiseksi ilman, että luokan käyttäjän täytyy tietää luotavan luokan konkreettista toteutusta.

Abstrakti tehdas -suunnittelumalli (Abstract Factory) tarjoaa rajapinnan yhteenkuuluvien tai toisistaan riippuvien olioiden luontiin [20]. Kuva 2.5 esittää Abstrakti tehdas -suunnittelumallin osat esimerkin avulla. *Abstrakti tehdas* (AbstractFactory) määrittää luontioperaatiot *tuotteille* A ja B (createProductA ja createProductB). Abstraktin *tehtaan konkreettiset toteutukset* (ConcreteFactory1 ja ConcreteFactory2) tarjoavat toteutukset kyseisille luontioperaatioille. Abstraktit tuotteet (AbstractProductA ja AbstractProductB) tarjoavat yhteiset kantaluokat tehtaan avulla luotaville erityyppisille tuotteille. Konkreettiset tuotteet (ProductA1 ja ProductB1 sekä ProductA2 ja ProductB2) luodaan niitä vastaavien tehtaiden avulla, ja ne ovat vastaavien abstraktien tuotteiden erikoistuksia. Sovellus (Client) käyttää erilaisia abstraktit kantaluokat toteuttavia tuotteita, jotka se luo tehtaan avulla.



Kuva 2.5. Abstrakti tehdas -suunnittelumalli. Mukailtu lähteestä [20].

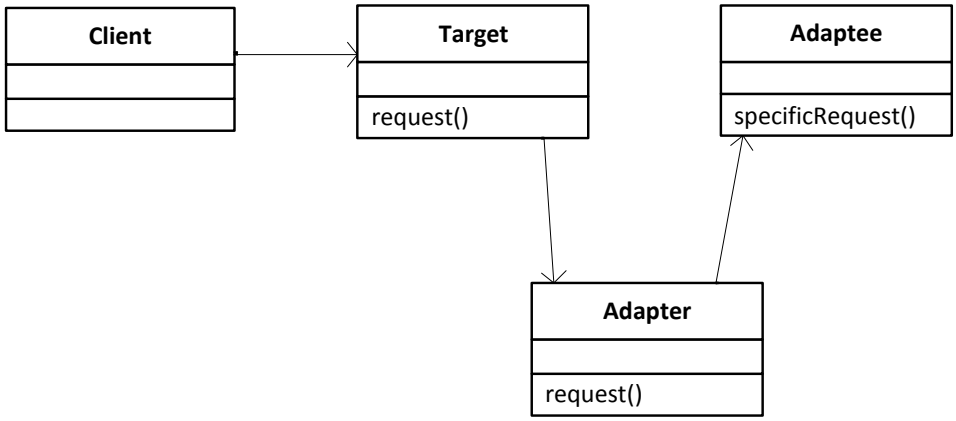
Abstrakti tehdas -suunnittelumallia voidaan käyttää luontiriippuvuuksien purkamiseen ohjelmistokehyksissä. Tällöin ohjelmistokehys pystyy luomaan erikoistuksien toteuttamia komponentteja abstraktin tehtaan avulla ilman, että sen täytyy tietää niiden konkreettisia toteutuksia. Erilaiset erikoistukset toteuttavat omat konkreettiset tehtaansa, joiden avulla niiden tarjoamien komponenttien toteutukset luodaan. [19]

Tehdasmetodi-suunnittelumallissa (Factory Method) määritellään luontioperaatio suoraan luokalle, joka käyttää luotavaa luokkaa. Luontioperaatio voidaan määritellä uudelleen tämän luokan aliluokassa, jolloin sen avulla voidaan luoda halutun tyyppisiä luokkia. Tehdasmetodi-suunnittelumalli soveltuu paremmin yksityiskohtaisen suunnittelun tasolle. [19]

2.3.4 Sovitin-suunnittelumalli

Sovitin-suunnittelumallia (adapter) käytetään muuttamaan luokan rajapinta sovelluksen ymmärtämään muotoon. Sen avulla saadaan luokat, joilla on keskenään yhtyeensopimattomat rajapinnat, toimimaan toistensa kanssa. [20]

Kuva 2.6 esittää Sovitin-suunnittelumallin osat ja niiden väliset suhteet. Asiakas (Client) pyytää palvelua (request) *kohteelta* (Target). Kohde määrittää sovellusaluekohtaisen rajapinnan, jota asiakas käyttää. *Sovitettava* (Adaptee) määrittää valmiin rajapinnan, joka tarjoaa tarvittavat palvelut, mutta ei ole suoraan yhteensopiva kohteen kanssa. *Sovitin* (Adapter) sovittaa sovitettavan rajapinnan kohteen ymmärtämään muotoon.



Kuva 2.6. Sovitin-suunnittelumalli. Mukailtu lähteestä [20].

Sovitin-suunnittelumallia käytetään esimerkiksi tapauksessa, jossa järjestelmä käyttää yleistä tietorakennekirjastoa tiettyjen tietorakenteiden toteutukseen. Jos kirjasto joudutaan jostain syystä vaihtamaan toiseen, uusi kirjasto tarjoaa todennäköisesti erilaiset rajapinnat tietorakenteiden toteutuksille. Sovitin-suunnnittelumallin avulla voidaan varautua tietorakennekirjastoon vaihtamiseen ja järjestelmän arkkitehtuuri voidaan suunnitella siten, että järjestelmä ei tule riippuvaiseksi tietyistä rajapinnoista. [19]

2.3.5 Ohjelmistokehykset

Koskimies & Mikkosen [19] mukaan ohjelmistokehys (framework) on ohjelmistorunko eli vaillinaisesti toteutettu ohjelmistotuote, jossa on aukkoja ennalta odotettuja täydennyksiä varten. Se on yleisesti käytetty olioperustainen tapa tuoterungon toteuttamiseksi jollekin ohjelmistojoukolle. Kehys uudelleenkäyttää komponenttien, luokkien ja/tai rajapintojen lisäksi myös ohjelmistojen arkkitehtuuria sekä perustoiminnallisuutta.

Ohjelmistokehys ei ole itsessään suorituskelpoinen ohjelmisto. Haluttu ohjelmisto saadaan täyttämällä kehyksen aukot koodilla, joka toteuttaa ohjelmiston vaatiman erityistoiminnallisuuden muuttamatta kehyksen tarjoamaa arkkitehtuuria. Tätä kutsutaan kehyksen *erikoistamiseksi* (specialization). Kehyksen sisältämät aukot ovat sen *laajennoskohtia* (hot spot). Kehys määrää, mitkä vaatimukset tuotekohtaisen koodin on täytettävä kussakin laajennoskohdassa. Näitä laajennoskohtia sekä niihin liittyviä vaatimuksia kutsutaan kehyksen *erikoistamisrajapinnaksi*. [19]

Tyypillisesti kehyksen erikoistamisrajapinta on huomattavasti perinteisen komponentin palvelurajapintaa mutkikkaampi. Palvelukutsut voivat kulkea molempiin suuntiin kehyksen ja tuotekohtaisen koodin välillä sekä laajennoskohdilla voi olla erilaisia riippuvaisuuksia keskenään. Esimerkiksi yhden laajennoskohdan täyttäminen tietyllä tavalla saattaa edellyttää toisen kohdan täyttämistä vastaavalla tavalla. Kehyksellä ja tuotekohtaisella koodilla on täten mutkikas vuorovaikutussuhde, johon osallistuu kummaltakin puolelta useita komponentteja kehyksen määrittämissä rooleissa. [19]

Tuotekohtainen koodi voidaan sitoa erikoistamisrajapinnassa kehyksen koodiin käyttäen erilaisia mekanismeja. Perinteisesti tähän käytetään periytymistä. Tällöin kehys

tarjoaa kantaluokan, ja tuotekohtainen luokka perii sen antaen samalla joillekin sen operaatioille uuden toteutuksen. Kehys voi luoda tuotekohtaisen aliluokan ilmentymiä tuntematta itse aliluokkaa, jos apuna käytetään tehdas-suunnittelumallia. Periytymisellä erikoistettavia kehyksiä kutsutaan *muunneltaviksi kehyksiksi* (white-box framework). Muita erikoistamismekanismeja ovat kehyksen tarjoamien rajapintojen toteuttaminen ja erilaiset parametrointimekanismit. [19]

Suunnittelumalleja käytetään usein osana kehysten arkkitehtuuria, koska myös useimpien suunnittelumallien tavoitteena on lisätä järjestelmän joustavuutta sekä muunneltavuutta. Kehyksen arkkitehtuuri voidaan usein kuvata pitkälle esittämällä, mitä suunnittelumalleja siinä on käytetty ja miksi. Tämä on usein paras tapa selittää kehyksen monimutkainen arkkitehtuuri. Suunnittelumalleja käytetään tyypillisesti sovelluksen ja tuotekohtaisen koodin rajapinnalla, mikäli suunnittelumallien käytöllä pyritään muunneltavuuteen. Lähes kaikissa suunnittelumalleissa onkin helposti nähtävässä yleinen, kehykseen kuuluva osa, ja vaihtuva, tuotekohtainen osa. [19]

3. MALLINNUSTYÖKALUT JA -YMPÄRISTÖT

Tässä luvussa kerrotaan suurten, maantieteellisesti hajautettujen organisaatioiden tarpeista mallinnustyökaluille. Lisäksi pohditaan perinteisten mallinnustyökalujen ja -ympäristöjen ongelmia kyseisessä kontekstissa sekä kerrotaan lyhyesti hajautetuista mallinnusympäristöistä. Lopuksi esitellään Tampereen teknillisellä yliopistolla kehitetty hajautettu mallinnus- ja työkaluympäristö – Trinity.

3.1 Mallinnustyökalut globaalien organisaatioiden kannalta

Suuret ja monimutkaiset järjestelmät tehdään yhteistyössä useiden ihmisten kesken, jotka voivat olla myös eri organisaatioista. Tällöin myös järjestelmien mallintamiseen osallistuu suuri joukko ihmisiä. Mallinnustyötä voidaan toki pilkkoa pienempiin osiin, ja jakaa osia yksittäisille henkilöille, mutta varsinkin kriittisimmät osat mallinnetaan useamman ihmisen yhteistyönä.

Suurissa kansainvälisissä organisaatioissa työntekijät ovat jakaantuneet eri puolilla maapalloa sijaitseviin toimipisteisiin. Eri toimipisteissä työskentelevien ihmisten pitää välillä päästä työskentelemään toistensa kanssa sekä tarkastelemaan muiden tekemiä malleja. Edes saman tiimin jäsenet eivät aina pääse suunnittelemaan saman pöydän äärelle, sillä osa henkilöistä saattaa matkustella paljon, ja jotkut saattavat tehdä etätöitä.

Suurissa järjestelmissä on jopa satoja analysoitavia ominaisuuksia. Ominaisuuksilla on riippuvuuksia toisiinsa ja päällekkäisyyksiä toistensa kanssa. Analysoitavaa on paljon, ja työ joudutaan jakamaan useiden henkilöiden kesken. Tällöin he käsittelevät samoja tietoja yhtä aikaa. Tämän vuoksi käsiteltävien tietojen pitää olla ajan tasalla sekä samoja tietoja pitää pystyä muokkaamaan yhtä aikaa.

3.1.1 Perinteiset mallinnustyökalut

Perinteisissä tiedostopohjaisissa mallinnusympäristöissä mallinnustyötä tehdään yhdellä tietokoneella kerrallaan, ja tiedostot voidaan lähettää muualle jatkokäsittelyä varten. Tällainen toimintatapa ei ole kovin joustava eikä sovellu hyvin maantieteellisesti hajautettuun organisaatioon.

Jotkut tiedostopohjaiset mallinnusympäristöt, kuten Power Designer [22], tarjoavat keskitetyn säiliön (repository) tiedostojen tallentamiseen ja hakemiseen. Mallit tai niiden osat lukitaan, kun ne haetaan säiliöstä muokattavaksi, jotta tietojen yhtäaikainen muuttaminen ei aiheutaisi konflikteja. Muut käyttäjät pystyvät lukemaan lukittuja malleja, mutta niiden sisältämien tietojen muuttaminen on estetty. Lukot vapautetaan,

kun mallit tallennetaan takaisin säiliöön. Keskitetyn säiliön käyttäminen helpottaa tiedostojen jakamista muiden käyttäjien välillä, mutta ei kuitenkaan mahdollista mallien samanaikaista käsittelyä.

Mallinnusympäristöt sisältävät yleensä pienen joukon erilaisia, pääosin graafisia, työkaluja, joilla on tarkoitus työskennellä määrätyllä tapaa, ja ne pakottavat käyttäjät tiettyihin työskentelytapoihin. Mallinnustyökalut eivät esimerkiksi tue erityisesti ohjelmistojen määrittelyvaiheessa tarvittavia epäformaaleja kuvauksia, vaan pakottavat mallit käytetyn mallinnuskielen sääntöjen mukaisiksi. Ihmiset käyttävät näistä syistä erilaisia tekstinkäsittely- ja taulukkolaskelmaohjelmia suunnittelun ja määrittelyn apuna. Näistä epäformaaleista kuvauksista tehdään myöhemmässä vaiheessa formaalit mallit, ja tämän jälkeen alkuperäiset kuvaukset jäävät irrallisiksi dokumenteiksi, jotka eivät ole mallien kanssa ajan tasalla.

3.1.2 Hajautetut ja integroidut mallinnusympäristöt

Hajautetussa mallinnusympäristössä samaa mallia ja jopa samaa näkymää voidaan käsitellä useasta paikasta samanaikaisesti. Mallien tieto on tallessa esimerkiksi tietokannassa, joka mahdollistaa tietojen rinnakkaisen käsittelyn ilman, että malleja täytyy lukita. Malliin tehtävät muutokset näkyvät muille lähes välittömästi. Tämä mahdollistaa mallintamisen yhteistyössä muiden ihmisten kanssa riippumatta heidän sijainnistaan.

Mallinnustyö on iteratiivista ja sisältää paljon erilaisia vaiheita. Projektin alkuvaiheessa työ on luonnostelevaa ja hahmottelevaa, kun taas myöhemmässä vaiheessa tarvitaan täydellisiä ja tiukasti käytetyn mallinnuskielen mukaisia malleja. Yksi työkalu ei ole yleensä paras joka tarkoitukseen, vaan työt kannattaa tehdä siihen kulloinkin parhaiten soveltuvalla työkalulla. Mallien tietojen pitää olla siirrettävissä työkaluista toisiin, jotta niitä voidaan käsitellä eri työkaluilla eri vaiheissa. Mallinnustietoa ei saa missään tapauksessa kadottaa. Lisäksi muunnokset työkalujen välillä pitää automatisoida, jotta työkaluympäristöstä saadaan käyttökelpoinen.

3.2 Trinity-mallinnusympäristö

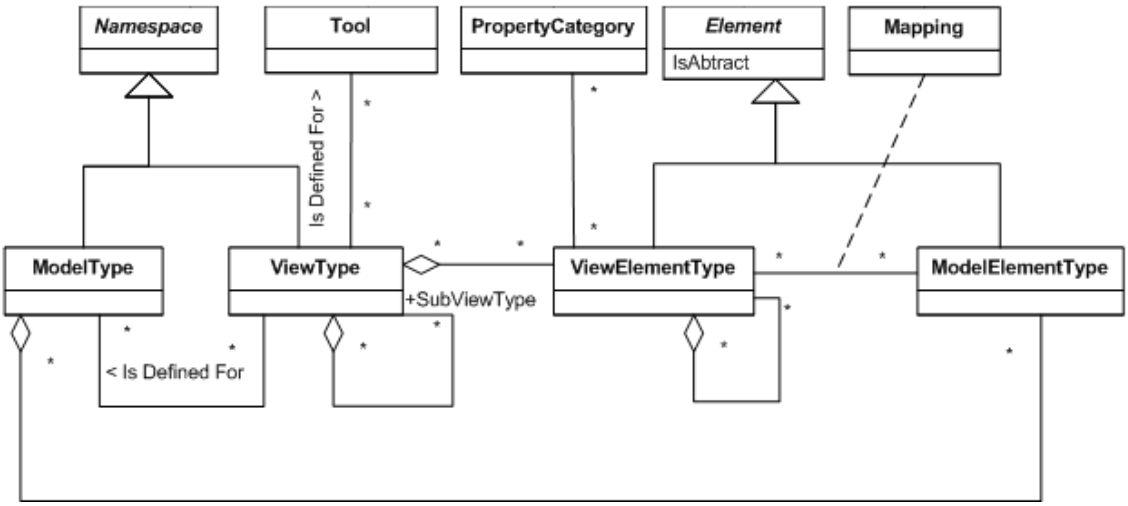
Trinity on Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella kehitetty hajautettu tietokantapohjainen mallinnus- ja työkaluympäristö, joka on suunniteltu ratkaisemaan muun muassa aliluvussa 3.1. selostettuja ongelmia. Trinitylle on määritelty oma metametamalli, johon ympäristön mallinnuskielet perustuvat. Metametamallia käyttäen on määritelty muun muassa OMG:n määrittelemä UML 2.3:n mallinnuskieli ja siihen liittyvät näkymätyypit [6]. Trinityyn on lisäksi kehitetty agenttiarkkitehtuuri [23,14], joka helpottaa erilaisten työkalujen integrointia ympäristöön sekä mahdollistaa työkalujen välisen yhteistoiminnan.

Trinityn yhtenä pääideana on tukea tehtävää työtä yleisesti käytössä olevan työkaluorientoituneen tavan sijaan, jossa työkalut pakottavat tekemään asiat tietyllä tapaa. Trinity ei ole itsessään mallinnustyökalu tai muukaan CASE-työkalu, vaan se tarjoaa mekanismit, joiden avulla ympäristöön voidaan joustavasti rakentaa ja integroida erilai-

sia työkaluja. Se muodostaa täten työkaluympäristön, joka voidaan räätälöidä erilaisia tarpeita ja työtapoja varten.

3.2.1 Trinityn tietomalli ja sen metatasot

Trinityn metametamallin avulla määritellään muun muassa ympäristön mallinnuskielet ja ympäristöön kuuluvat työkalut. Liite 1 kuvaa Trinityn metametamallin, josta Kuva 3.1. sisältää otteen. Metametamallin keskeisin elementti on abstrakti *Thing-elementti*, josta melkein kaikki muut metametamallin elementit periytyvät. Sen ominaisuuksia ovat nimi ja kuvaus. ModelType-elementti edustaa ympäristöön määriteltyä *mallityyppiä*. Se sisältää joukon mallinnuskieleen määriteltyjä *mallielementtityyppejä* (ModelElementType). Tool-elementti puolestaan edustaa ympäristöön kuuluvaa *työkalua*. Työkaluille on määritelty *näkymätyyppejä* (ViewType), jotka sisältävät *näkymäelementtityyppejä* (ViewElementType). Näkymäelementtityyppi määritellään mallielementtityypille. Näkymäelementtityyppiin liittyy PropertyCategory-elementti, joka kuvaa näkymäelementtityypille kuuluvaa työkalukohtaista tietoa liittyen elementin esittämiseen työkalussa.



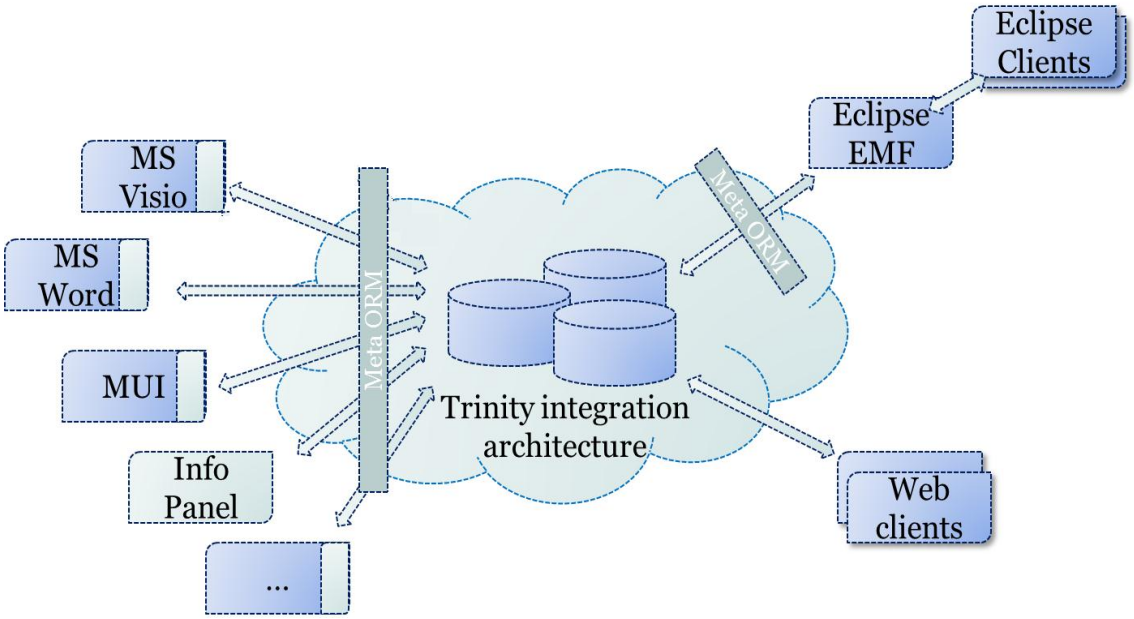
Kuva 3.1. Ote Trinityn metametamallista.

Trinityyn on määritelty muun muassa Visio-mallinnustyökalu työkaluksi ja mallityyppi UML-luokkakaavioille. Mallityyppi sisältää OMG:n UML-luokkakaaviolle määrittämät elementit [6] mallielementtityyppeinä. Mallityypille on lisäksi määritelty esimerkiksi luokkakaavionäkymätyyppi, joka on tarkoitettu käytettäväksi Visio-laajennuksella.

3.2.2 Ympäristö ja palvelut

Kuva 3.2 näyttää yleiskuvan Trinitystä ja sen eri osista. Trinityn keskeisimpänä osana on hajautettu tietokanta, johon tallennetaan mallinnuskielet, mallit sekä muu ympäristölle tarpeellinen tieto. Ympäristöön on toteutettu olio-relaatio-muunnokset (Meta ORM) tekevät tietokantakomponentit Microsoft .NET -sovelluskehykselle ja Javalle. Tässä

työssä keskitytään .NET-kehyksellä toteutettuun osaan, koska se on oleellisin Excel-laajennuksen kannalta.



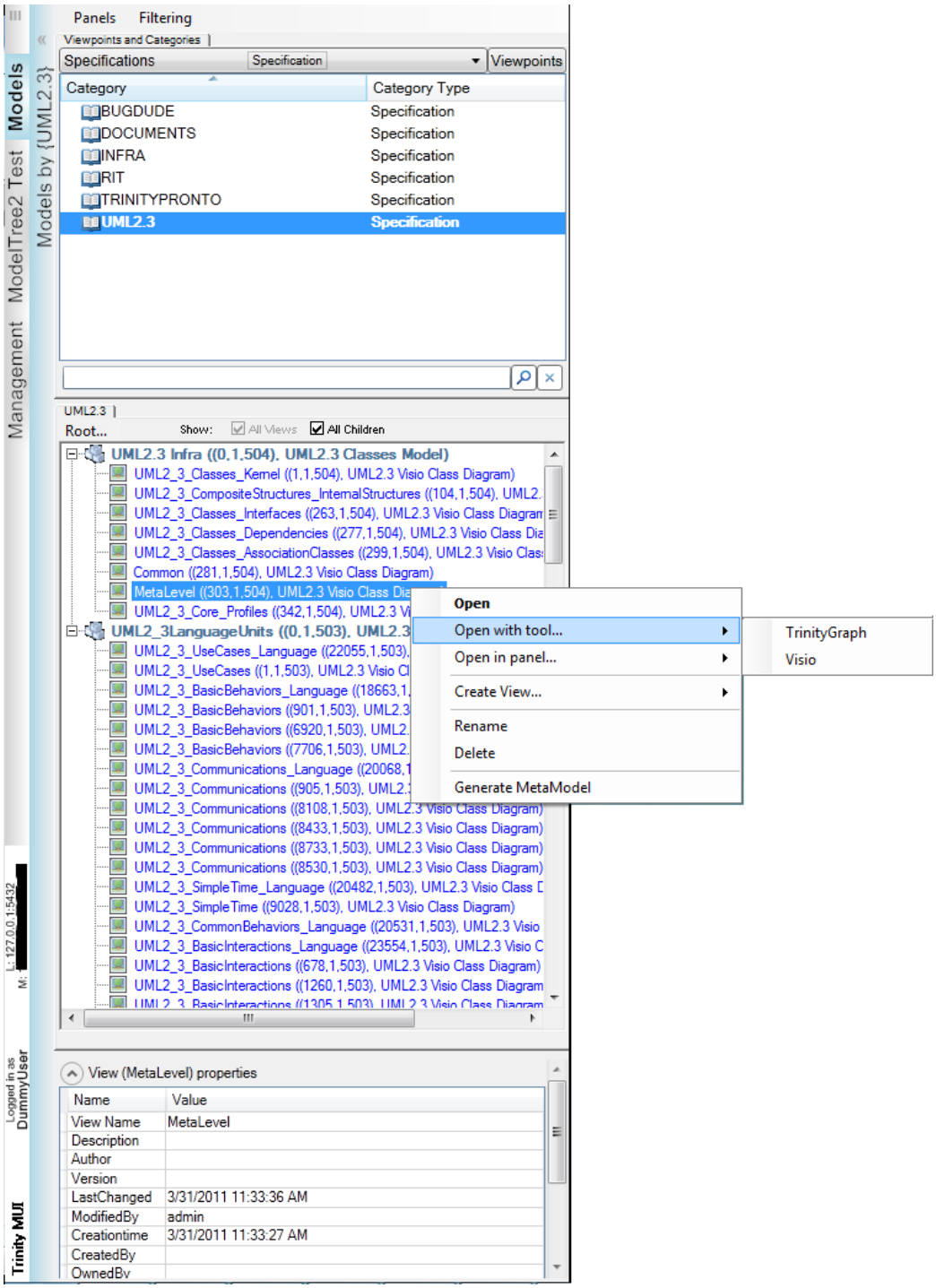
Kuva 3.2. Trinityn yleiskuva. Mukailtu Jari Peltosen tekemästä Trinityn esittelymateriaalista.

Tietokantakomponentin tehtävänä on muuntaa tietokannassa relaatiomuodossa oleva tieto ohjelmoijille paremmin sopivaan oliomuotoon. Tietokantakomponenttiin on toteutettu myös monia muita ohjelmoijia helpottavia ominaisuuksia. Se tarjoaa esimerkiksi transaktioidenhallintamekanismit sekä tuen suoritettujen transaktioiden kumoamiselle ja uudelleen tekemiselle. Tietokantakomponentti tiedottaa mallien ja näkymien tietojen muuttumisesta tapahtumapohjaisesti kaikille muutoksista kiinnostuneille.

.NET-sovelluskehyksellä toteutettu osio sisältää tietokantakomponentin lisäksi hallintakäyttöliittymän, Microsoft Visio ja Microsoft Word -laajennukset sekä informaatiopaneelin (Info Panel), jolla voidaan näyttää elementtien tietoja työkaluissa. Lisäksi työkalujen integrointiin ja yhteistoimintaan on toteutettu agenttiarkkitehtuuri. Näiden lisäksi Trinity sisältää työn kirjoitushetkellä Javalle tehtyjä komponentteja (Eclipse EMF, Eclipse Clients) sekä erilaisia web-pohjaisia käyttöliittymiä (Web clients) mallien tietojen käsittelyyn.

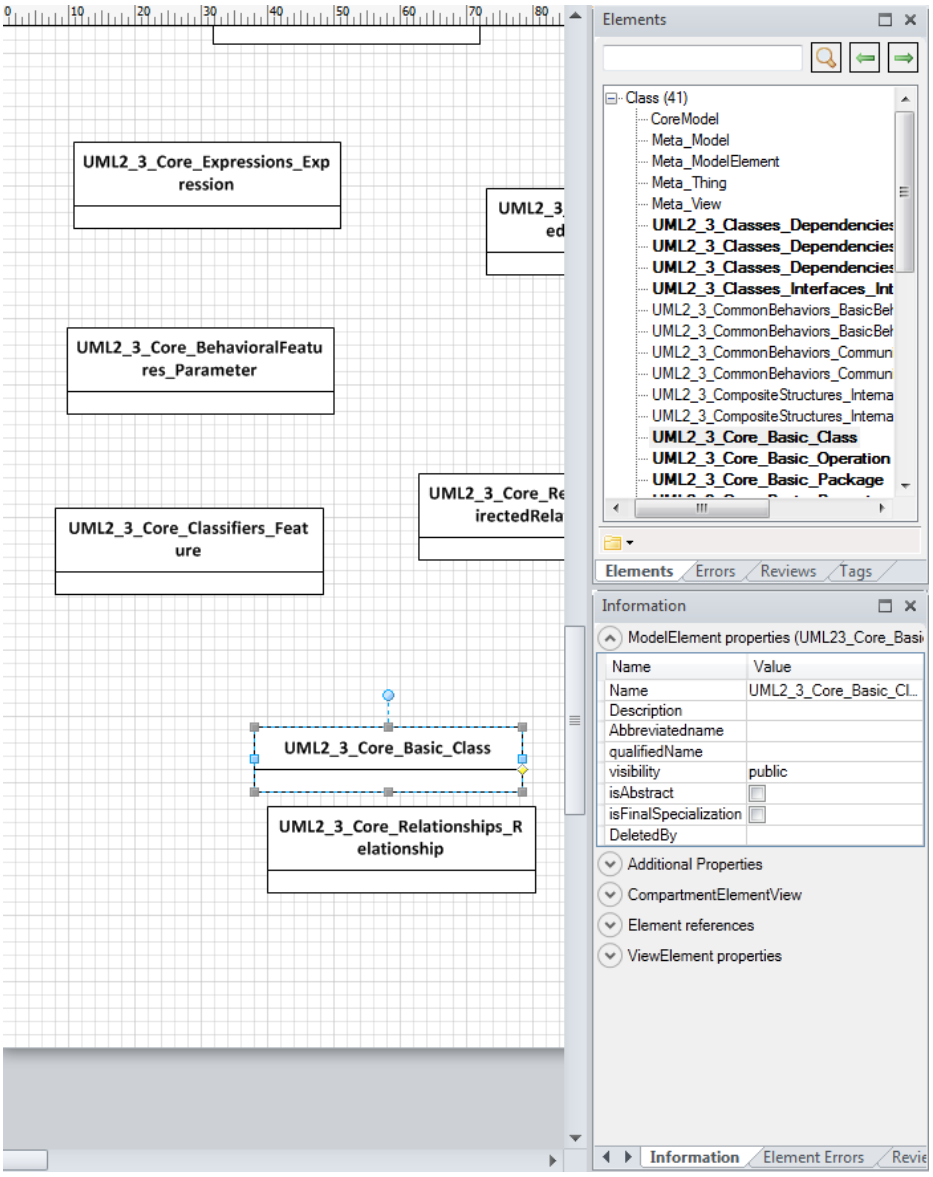
Hallintakäyttöliittymä (Management User Interface, MUI) [25] on tarkoitettu ympäristön mallien ja näkymien hallintaan. Sen avulla voidaan luoda uusia malleja ja näkymiä sekä järjestää niitä eri tavoin. Lisäksi se toimii ympäristön työkalujen keskipisteenä, josta näkymät avataan käsiteltäväksi eri työkaluihin (Kuva 2.1).

Ensimmäinen Trinityyn integroitu mallinnustyökalu toteutettiin Microsoft Visio -laajennuksena [26,27]. Microsoft Visio on yleisesti käytössä oleva työkalu erilaisten kaavioiden piirtämiseen. Kirjoitushetkellä Visio-laajennukseen on määriteltynä näkymätyypit UML 2.3:n määrittelemiin kaaviotyyppeihin.



Kuva 3.3. Näkymän avaaminen työkaluun hallintakäyttöliittymästä.

Informaatiopaneeli on itsenäinen komponentti, joka tarjoaa toiminnot lomakemuotoisen mallitiedon lukemiseen ja muokkaamiseen. Paneeli voidaan liittää osaksi ympäristön työkalujen käyttöliittymiä (Kuva 3.4). Työkalut esittävät itse elementtien näkymäinformaation. Mallielementeillä saattaa kuitenkin olla niin paljon erilaisia ominaisuuksia, että niitä kaikkia ei ole sopivaa esittää näkymäelementeissä. Informaatiopaneelin avulla on tarkoitus näyttää työkalussa valittuna olevien mallielementtien tiedot.



Kuva 3.4. Informaatiopaneeli osana Trinityn Visio-laajennuksen käyttöliittymää.

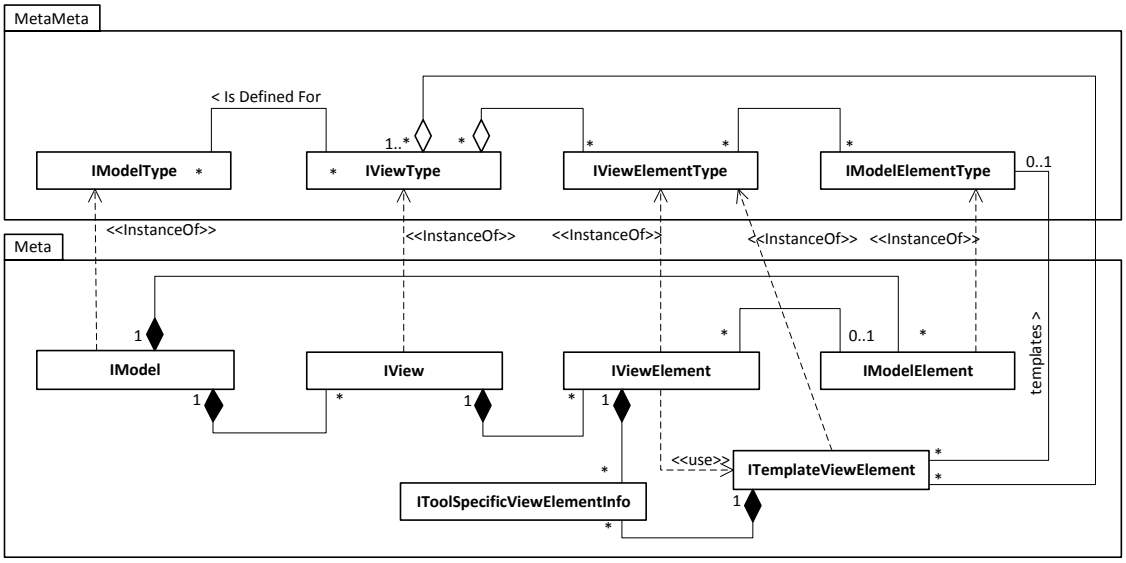
Informaatiopaneeli sisältää avoimena olevan mallin kaikki mallielementit näyttävän puunäkymän ja valittujen elementtien tietoja näyttävän informaatioikkunan. Puunäkymästä voidaan muun muassa etsiä elementtejä sekä raahata ja pudottaa niitä hiirellä työkaluihin, jolloin niille luodaan viittaava näkymäelementti avoimeen näkymään.

3.2.3 Laajennusmekanismit uusien työkalujen integroimiseksi

Uusien työkalujen integrointi Trinityyn on yritetty tehdä mahdollisimman helpoksi. Ympäristön tietomalliin voidaan lisätä uusia työkaluja ja työkaluille voidaan määritellä erilaisia näkymätyyppejä. Näkymätyypin ja valitun mallielementintyypin välille luodaan suhde, jonka jälkeen hallintakäyttöliittymästä voidaan luoda uusia näkymätyypin mukaisia näkymiä siihen liittyvän mallityypin tyyppisiin malleihin.

Kuva 3.5 näyttää otteen tietokantakomponentin käyttörajapinnasta, jossa näkyy Trinityn laajentamiseen läheisesti liittyvät käsitteet. N*äkymäelementit* (IViewElement)

ja *Template-elementit* (ITemplateViewElement) ovat yhteen ympäristön työkaluun liittyvän näkymäelementtityypin (IViewElementType) ilmentymiä. Näkymätyypeille voidaan määritellä joukko template-elementtejä, jotka määrittävät näkymätyypin mukaisessa näkymässä (IView) käytössä olevat elementit. Template-elementti määrittää sitä käyttävien näkymäelementtien ulkoasun sekä niihin mahdollisesti liittyvän *mallielementin* (IModelElement) tyypin.



Kuva 3.5. Ote tietokantakomponentin käyttörajapinnasta. Trinityn laajentamiseen läheisesti liittyvät käsitteet.

Työkalukohtaisille näkymäelementtien tiedoille (IToolSpecificViewElementInfo) voidaan toteuttaa operaatiot tietokannasta lataamista ja sinne tallentamista varten. Tietokantakomponentti sisältää plugin-mekanismin, jonka avulla kyseiset toiminnot voidaan rekisteröidä sille. Rekisteröinnin jälkeen se osaa muun muassa ladata tarvittavat elementtien työkalukohtaiset tiedot näkymän avaamisen yhteydessä sekä suorittaa niille tarvittavat toimenpiteet operaatioiden kumoamisen ja uudelleen tekemisen yhteydessä.

Lisäksi integroitavalle työkalulle toteutetaan tarvittavat komponentit Trinityn agenttiarkkitehtuuriin, jonka jälkeen se pystyy reagoimaan hallintakäyttöliittymältä saapuviin pyyntöihin, kuten työkalun käynnistämiseen ja näkymien avaamiseen. Agenttiarkkitehtuuriin on myös mahdollista määritellä uusia agentteja, joiden avulla eri työkalut saadaan toimimaan yhteistyössä keskenään halutulla tavalla.

4. TAULUKKOLASKENTAOHJELMAT OHJELMISTOTUOTANNON APUVÄLINEENÄ

Tässä luvussa kerrotaan taulukoiden käyttämisestä ohjelmistotuotannossa sekä pohditaan erillisten taulukkolaskentaohjelmien käytöstä aiheutuvia ongelmia raportointi- ja mallinnuskäytössä. Lopuksi esitellään joukko vaatimuksia taulukkolaskentaohjelman laajentamiseksi raportointi- ja mallinnustyökaluksi Trinityyn.

4.1 Taulukoiden käyttö ohjelmistotuotannossa

Taulukoitua tietoa voidaan ajatella tietyssä mielessä graafisen ja tekstimuotoisen esitystavan yhdistelmänä, koska taulukoilla on tietty rakenne, mihin on yhdistetty tekstiä. Taulukoiden rakenteen avulla niissä voidaan kuvata tietojen välisiä riippuvuuksia, jotka saadaan selville nopealla vilkaisulla. Esimerkkinä tästä voidaan mainita päätöstaulut, joissa voidaan kuvata monimutkaisia päätöstilanteita havainnollisemmin kuin esimerkiksi loogisilla lausekkeilla [28].

Ohjelmistotuotannossa käytetään paljon taulukoita erilaisten tietojen esittämiseen. Yksi syy tälle on, että taulukot ovat luontevin esitysmuoto tietyn tyyppisten tietojen esittämiseen. Ne sopivat erityisen hyvin tietojen ristiintaulukointiin esimerkiksi erilaisten matriisien tekemiseen. Lisäksi suurten tietomäärien esittäminen graafisesti elementeillä ei onnistu välttämättä yhtä hyvin kuin taulukoituna. Taulukko 4.1 näyttää esimerkin tiedoista, jotka soveltuvat hyvin esitettäväksi taulukkomuodossa. Se sisältää otteen tietohakemistosta, jossa esitetään käsitteitä, niiden ominaisuuksia ja kuvauksia.

Taulukko 4.1. Esimerkki käsitteiden esittämisestä tietohakemistossa. Mukailtu lähteestä [28].

Käsitteen nimi	**Ominaisuudet**	**Kuvaus**
Työntekijä	Sukunimi + Etunimi + Hetu + Sukupuoli	Työntekijän perustiedot
Sukunimi	0{M}40	Työntekijän nykyinen sukunimi
Etunimi	0{M}40	Työntekijän etunimi
Hetu	0{M}11	Työntekijän henkilötunnus. Oikeellisuus on tarkistetaan tarkistemerkin perusteella.
Sukupuoli	'M'|'N'	Automaattisesti henkilötunnuksen perusteella

Ohjelmistoprosessi alkaa yleensä vaatimusten keräämisestä [28]. Kerätyt vaatimukset on luontevaa esittää taulukossa, missä kuvataan esimerkiksi vaatimusten nimet, kuvaukset ja prioriteetit. Tällaisten tietojen syöttäminen on huomattavasti nopeampaa taulukoihin kuin graafisesti, koska tiedot voidaan kirjoittaa näppäimistön avulla. Edellä mainittujen tapojen lisäksi esimerkiksi simuloinneista saatuja tuloksia ja erilaisia analyysejä on luontevaa esittää taulukoituina raportteina esimerkiksi testausta varten. Muita käyttötapoja ovat lisäksi testitapausten, poikkeusten, parametrien ja viestien listaus.

4.1.1 Raportointikäytön ongelmat

Taulukkolaskentaohjelmia – kuten muitakin toimisto-ohjelmistoja – käytetään paljon erilaisten raporttien tekemiseen. Mallien tiedoista luodaan taulukkomuotoisia raportteja, esimerkiksi tietohakemistoja. Suurin osa perinteisistä CASE-työkaluista tarjoaa mahdollisuuden teksti- ja taulukkomuotoisten raporttien tekemiseen. Esimerkiksi PowerDesigner [22] ja MagicDraw [29] tarjoavat mahdollisuuden muun muassa RTF- ja HTML-muotoisten raporttien generoimiseen mallien tiedoista. Raportteihin voidaan generoida myös taulukoita, jotka sisältävät elementtejä, niiden ominaisuuksia sekä niihin liittyviä toisia elementtejä.

Perinteisten CASE-työkalujen ongelmana on, että generoidut raportit ovat staattisia eivätkä malleihin tehdyt muutokset päivity niihin automaattisesti. Tämän vuoksi raportit joudutaan generoimaan uudestaan aina muutosten jälkeen, mikä aiheuttaa turhaa työtä. Toinen ongelma on, että tietoja ei voida muokata suoraan raportista. Esimerkiksi raporttia tarkasteltaessa löytyneen virheen korjaaminen malliin vaatii mallin avaamista käsiteltäväksi toiseen työkaluun ja korjausten tekemistä siellä.

4.1.2 Mallinnuskäytön ongelmat

Taulukkolaskentaohjelmia käytetään paljon myös ohjelmistojen mallinnuksessa, varsinkin määrittelyvaiheessa, koska perinteiset graafiset mallinnustyökalut eivät ole tarpeeksi joustavia määrittelyn alkuvaiheessa syntyville nopeasti muuttuville ja epäformaaleille malleille. Taulukkolaskentaohjelmat soveltuvatkin hyvin esimerkiksi määrittelyvaiheen käsitteiden ja niiden ominaisuuksien luetteloimiseen. Elementtien ominaisuuksien – erityisesti pitkien kuvausten – syöttäminen näppäimistön avulla on huomattavasti nopeampaa kuin graafisissa mallinnustyökaluissa yleisesti käytössä oleva elementtien pudottaminen ja näppäily hiirellä [30].

Määrittelyn saavuttaessa tarpeellisen kypsyysasteen epäformaalien kuvausten pohjalta tehdään yleensä formaalimmat mallit käyttäen esimerkiksi UML-luokkakaavioita. Mallien muuntaminen graafisiksi kaavioiksi aiheuttaa turhaa mekaanista työtä, koska CASE-työkalut eivät tarjoa mahdollisuutta automaattiseen mallien generointiin. Muunnoksissa saatetaan myös menettää tietoa, koska käytetty graafinen mallinnustyökalu ei välttämättä tue kaikkia taulukkolaskentaohjelmassa käytettyjä epäformaaleja kuvaustapoja. Alkuperäiset taulukoidut tiedot eivät tule suoraan osaksi mallia, vaan jäävät irrallisiksi dokumenteiksi. Muutokset täytyy tehdä erikseen sekä graafi-

siin kaavioihin että taulukoihin, jos molempien halutaan pysyvän ajan tasalla. Tämä aiheuttaa jälleen kerran turhaa työtä. Lisäksi tietojen ollessa ristiriidassa keskenään muodostuu riskiksi se, että ei pystytä varmistamaan kummat tiedot ovat oikeita.

4.2 Vaatimukset taulukkolaskentaohjelman laajentamiseksi raportointi- ja mallinnustyökaluksi Trinityyn

Trinityyn haluttiin lisätä mahdollisuus erilaisten taulukkomuotoisten raporttien generointiin valituista malleista ja näkymistä eri sidosryhmien tarpeita varten. Raportointiin kannattaa käyttää tavallisia toimisto-ohjelmia, koska ne ovat tehty tällaista tarkoitusta varten, ja käyttäjät ovat tottuneet käyttämään niitä. Raporttien halutaan päivittyvän automaattisesti mallin tietojen muuttuessa, jotta vältytään ylimääräisiltä raporttien generoinneilta.

Raportteihin tehtävät muutokset halutaan päivittää suoraan malliin. Esimerkiksi käyttäjän huomatessa raportissa virheen hän voisi tehdä korjauksen suoraan siihen ilman, että hänen täytyisi ensin avata malli toiseen työkaluun, tehdä muutokset sinne ja generoida raportti uudestaan.

Joissakin tapauksissa myös mallien sisältämien tietojen käsittely on luontevinta tehdä taulukkomuodossa, joten uuden työkalun täytyy mahdollistaa tämä. Trinityn ympäristö tarjoaa valmiina mekanismit tietojen muutosten ilmoittamisesta, mutta tietoja pitää pystyä muuttamaan myös suoraan taulukkomuotoisesta näkymästä, ja muutosten pitää tallentua välittömästi Trinityn tietokantaan.

4.2.1 Raportointiominaisuudet

Trinityn mallityypeille voidaan luoda näkymätyypeiksi erilaisia raporttipohjia, jotka määrittävät niitä käyttävien raporttien ulkoasun sekä raporttiin generoitavien elementtien tyypit, näytettävät ominaisuudet ja niihin liittyvät toiset elementit. UML-luokkamallin mallityyppiin voidaan määritellä raporttipohja esimerkiksi tietohakemistolle, jossa näytetään taulukoituna UML-luokkia sekä niiden nimet ja kuvaukset. Jokaisesta luokasta näytetään myös niihin liitetyt toiset elementit kuten, stereotyypit, attribuutit ja operaatiot. Lisäksi näytetään luokkiin periytymissuhteilla liitetyt luokat sekä assosiaatiot ja niiden toisiin päihin kiinnitetyt luokat.

Esimerkkitapaus 1 esittää esimerkin tietohakemisto raportin tekemisestä Trinityn mallista. Lisäksi siinä kuvataan raporttiin generoitavien elementtien suodattaminen valittujen kriteerien avulla sekä lähdemalliin tehtyjen muutosten päivittyminen raporttiin.

Käyttäjä haluaa generoida tietohakemisto-raportin UML-luokkamallista, mikä sisältää Trinityn Visio-laajennuksella tehtyjä näkymiä. Hän luo malliin uuden tietohakemistotyyppisen näkymän, mille hän asettaa lähteeksi edellä mainitun mallin. Hän haluaa raportin sisältävän kaikki viimeisen kuukauden aikana itse luomansa luokat. Lisäksi hän haluaa luokkiin mukaan operaatioista vain ne, joiden näkyvyysmääre on julkinen ja joihin ei liity yhtään parametria. Hän lisää raporttiin luokkiin kohdistuvat suodattimet luokan tekijälle ja luontiajankohdalle. Lisäksi hän lisää luokkiin liittyville operaatioille suodattimet näkyvyysmäärettä ja operaatioihin liittyviä parametreja varten. Lopuksi käyttäjä hyväksyy raportin asetukset, jonka jälkeen raportin sisältö generoidaan.

Tämän jälkeen käyttäjä avaa raportin lähdemallissa sijaitsevan Visio-kaavion käsiteltäväksi. Hän lisää kaavioon uuden luokan ja sille uuden operaation. Hän asettaa operaation näkyvyysmääreen yksityiseksi. Lisäksi hän siirtää toisen käyttäjän tekemästä luokasta yhden attribuutin luomalleen uudelle luokalle. Tehdyistä muutoksista generoituu automaattisesti raporttiin uusi näkymäelementti luodulle luokalle, mutta sille ei lisätä parametria, koska asetettu suodatin suodattaa sen pois. Myös siirretylle attribuutille generoidaan uusi näkymäelementti luokan lapsielementiksi.

Esimerkkitapaus 1. Raportin luominen mallista.

Taulukko 4.2 listaa raportointitoimintojen vaatimukset. Trinityssä olevista malleista ja näkymistä on pystyttävä generoimaan taulukkomuotoisia raporttinäkymiä. Uusia raportteja pitää pystyä luomaan Trinityn hallintakäyttöliittymästä haluttuun malliin.

Taulukko 4.2. Raportoinnin vaatimukset.

Tunniste ja nimi	**Kuvaus**
V01. Raporttinäkymän generointi malleista ja toisista näkymistä	Malleista ja näkymistä on pystyttävä generoimaan valitun raporttipohjan mukaisia raportteja.
V02. Elementtien suodattaminen	Raporttinäkymään generoitavia elementtejä on pystyttävä suodattamaan erilaisten kriteerien perusteella.
V03. Muutosten automaattinen päivittyminen raporttiin	Generoidun raportin pitää päivittyä automaattisesti, kun lähdemalleihin tai -näkymiin tehdään muutoksia.

Raporttinäkymään generoitavia elementtejä on pystyttävä suodattamaan erilaisten kriteerien perusteella, jotka liittyvät esimerkiksi elementtien ominaisuuksilla oleviin arvoihin, kuten niiden nimiin, tai elementtiin liittyviin toisiin elementteihin. Myös elementin mahdollisia lapsielementtejä on pystyttävä suodattamaan. Suodattimia pitää pystyä asettamaan enemmän kuin yksi, ja niitä pitää pystyä myös muokkaamaan sekä poistamaan. Suodattimiin tehtyjen muutosten jälkeen raportit päivitetään vastaamaan muuttunutta tilannetta.

Raporttien pitää päivittyä, kun lähdemalleihin tai -näkymiin tehdään muutoksia. Esimerkiksi mallielementtien ominaisuuksien arvojen, kuten nimen tai kuvauksen, muutosten pitää päivittyä automaattisesti raportteihin. Lisäksi uusien mallielementtien luomisen jälkeen luodaan mallielementeille näkymäelementit automaattisesti generoituun raporttiin. Myös elementtien välisten suhteiden muuttuminen päivitetään generoituihin näkymäelementteihin. Mallien tietojen lisäksi myös näkymään tehtyjen muutosten, esimerkiksi värien, fonttien sekä elementtien sijaintien vaihtamisten, pitää päivittyä automaattisesti raportteihin.

4.2.2 Mallinnusominaisuudet

listaa mallinnustoimintojen vaatimukset. Taulukkolaskentaohjelman laajennuksen tulee ensisijaisesti tukea työtä Trinityn periaatteiden mukaisesti. Työkalulla mallintamiseen ei saa lisätä tarpeettomia tai toisteisia työvaiheita. Mallinnustoiminnot tulee tehdä ensisijaisesti taulukkolaskentaohjelman omilla toiminnoilla, jotta työkalun käyttö olisi mahdollisimman tuttua ja intuitiivista ohjelmaa aiemmin käyttäneille. Laskentataulukoita on pystyttävä muokkaamaan samalla lailla kuin ohjelmaa tavallisesti käytettäessä. Esimerkiksi tiedon muokkaus-, siirto- ja muotoilutoimintojen on käyttäydyttävä kuten ohjelman normaalikäytössä. Lisäksi tehtyjä toimintoja on pystyttävä kumoamaan ja tekemään uudelleen. Myös käyttöliittymän tulee säilyä ennallaan ja siihen lisätään ainoastaan mallinnustoiminnot, joita ei voi tehdä suoraan ohjelman omilla toiminnoilla, sekä Trinityn informaatiopaneeli elementtien tietojen näyttämistä varten.

Esimerkkitapaus 2. kuvaa esimerkin, jossa käyttäjä käsittelee mallia esimerkkitapaus 2:ssa generoidun raporttinäkymän kautta. Esimerkissä käyttäjä muuttaa mallielementtien tietoja, luo uusia mallielementtejä sekä navigoi lopuksi toiseen ympäristön työkaluun käsittelemään samoja tietoja.

Käyttäjä haluaa muokata mallin tietoja edellisen aliluvun esimerkkitapaus 1:ssä luodun raporttinäkymän kautta. Hän kirjoittaa luokille kuvaukset niille tarkoitettuihin kohtiin laskentataulukossa sekä muuttaa yhden luokan nimeä, koska aiempi nimi ei kuvannut tarpeeksi hyvin luokan tarkoitusta. Lisäksi hän huomaa, että mallista puuttuu tärkeä käsite, joten hän luo uuden UML-luokan ja antaa sille nimen sekä lisää sille kaksi attribuuttia.

Tämän jälkeen käyttäjä haluaa luoda uudelle luokalle näkymäelementin myös Visio-kaavioon. Hän valitsee yhden raporttiin generoiduista luokista ja navigoi siitä ensisijaiseen näkymäelementtiin. Trinityn Visio-laajennus käynnistyy, ja siihen avataan Visio-kaavio, jossa ensisijainen näkymäelementti sijaitsee. Hän kopioi raportista luodun luokan leikepöydälle ja liittää sen Visio-näkymään, jolloin sinne luodaan viittaava näkymäelementti luokalle.

Esimerkkitapaus 2. Mallintaminen työkalulla.

Taulukko 4.3. Mallinnustoimintojen vaatimukset.

Tunniste ja nimi	**Kuvaus**
V04. Ei ylimääräisiä vaiheita työhön	Työskentelyyn ei saa lisätä tarpeettomia tai toisteisia työvaiheita.
V05. Riittävä suorituskyky	Laajennuksen suorituskyvyn on oltava riittävä hyvän käyttökokemuksen säilyttämiseksi.
V06. Toimintojen kumoaminen ja uudelleen tekeminen	Kaikki toiminnot täytyy olla kumottavissa (undo) ja tehtävissä uudelleen kumoamisen jälkeen (redo).
V07. Mallielementin ominaisuuksien muokkaaminen	Mallielementin ominaisuuksia, kuten sen nimi ja kuvaus, pitää pystyä muuttamaan.
V08. Näkymän muotoilujen muutosten tallentuminen	Näkymän muotoilujen kuten väritysten, fonttien, rivien koon ja tietojen sijainnin muutosten pitää tallentua.
V09. Uusien elementtien luominen	Näkymään pitää pystyä luomaan uusia näkymätyyppiin määriteltyjä elementtejä.
V10. Viittaavien näkymäelementtien luominen	Olemassa oleville mallielementeille pitää pystyä luomaan viittaavia näkymäelementtejä laskentataulukkonäkymiin.
V11. Elementtien poistaminen	Elementtejä pitää pystyä poistamaan näkymästä.
V12. Elementtien kopiointi työkalussa	Elementtejä pitää pystyä kopioimaan työkalussa.
V13. Elementin kopiointi toisesta eri työkalusta	Elementtejä pitää pystyä kopioimaan ympäristön toisista työkaluista.
V14. Lapsielementtien lisääminen	Elementeille pitää pystyä lisäämään uusia lapsielementtejä.
V15. Lapsielementtien poistaminen	Elementeillä olevia lapsielementtejä pitää pystyä poistamaan.
V16. Lapsielementtien siirtäminen	Lapsielementtejä pitää pystyä siirtämään toiseen isäelementtiin tai omaksi päätason elementikseen
V17. Elementtien luominen valitusta alueesta	Laskentataulukossa olevaa tietoa pitää pystyä liittämään osaksi mallia. Käyttäjä voi luoda valinnaisesta alueesta haluamansa tyyppisen mallielementin.
V18. Tiedon sitominen mallielementtiin	Elementin alueella oleva tieto pitää pystyä sitomaan mallielementin ominaisuuteen.
V19. Navigointi ensisijaiseen näkymäelementtiin	Viittavasta näkymäelementistä pitää pystyä navigoimaan mallielementin ensisijaiseen näkymäelementtiin.
V20. Elementtien muutosten korostaminen	Elementteihin kohdistuva muutoksia pitää pystyä korostamaan.

Laajennuksen suorituskyvyn on oltava riittävä, jotta työkalun käytettävyys säilyisi hyvänä. Erityisesti käyttöliittymän vasteajan tulee olla riittäväni lyhyt, jotta työskentely ei häiriintyisi liikaa. Ideaalitapauksessa eroa työkalun tavalliseen käyttöön ei tulisi havaita. Pitkistä latausoperaatioista on ilmaistava käyttäjälle selkeästi, jotta hän ei odota välitöntä vastetta käyttöliittymältä.

Mallinnustoimintojen tulee toimia sekä raporttinäkymissä että muokattavissa näkymissä, joita ei ole sidottu lähdemalleihin tai -näkymiin. Mallielementtien ominaisuuksia on pystyttävä muuttamaan taulukkolaskentaohjelman omilla toiminnoilla. Lisäksi laskentataulukon muotoilut sekä tietojen sijainnit tulee tallettaa.

Näkymän kautta on pystyttävä luomaan uusia mallielementtejä. Niille luodaan näkymään ensisijainen näkymäelementti saman lailla kuin Trinityn Visio-laajennuksessa. Lisäksi olemassa oleville mallielementeille on pystyttävä luomaan viittaavia näkymäelementtejä raahaamalla ja pudottamalla niitä hiirellä informaatiopaneelista ja hallintakäyttöliittymästä.

Elementtejä on pystyttävä poistamaan tyhjentämällä kaikki elementin tiedot laskentataulukosta. Mikäli poistettava näkymäelementti on siihen liittyvän mallielementin ensisijainen näkymäelementti, myös mallielementti poistetaan.

Elementtejä pitää pystyä kopioimaan taulukkolaskentaohjelman oman kopioi-ja-liitä-toiminnon avulla sekä yhden näkymän sisällä että eri näkymien välillä. Uusien elementtien ulkoasut ja asetetut ominaisuudet kopioidaan alkuperäisiltä elementeiltä. Elementtejä on pystyttävä myös kopioimaan toisista työkaluista, esimerkiksi Visio-laajennuksesta, hiirellä raahaamalla ja pudottamalla sekä kopioimalla valitut elementit leikepöydälle ja liittämällä ne laskentataulukkoon. Kopioinnin pitää toimia myös toiseen suuntaan eli laskentataulukoista muihin Trinityn työkaluihin.

Elementeille pitää pystyä lisäämään toisia elementtejä lapsielementeiksi näkymätyyppien määritysten mukaisesti. Esimerkiksi tietohakemistonäkymätyypissä luokat voivat sisältää muun muassa attribuutteja ja operaatioita. Lapsielementtejä pitää myös pystyä poistamaan sekä siirtämään toiselle isä-elementille tai päätason elementiksi, jolla ei ole ollenkaan isä-elementtiä.

Näkymän tietoja on pystyttävä liittämään osaksi mallia. Esimerkiksi näkymään tekstimuodossa kirjoitettuja luokkien nimiä ja kuvauksia voidaan valita ja luoda niistä uusia luokkia. Valituista alueista luodaan uusille mallielementeille näkymäelementit. Näkymäelementtien sisältämiä tietoja on myös pystyttävä sitomaan mallielementtien ominaisuuksiin. Yllä olevan esimerkin tapauksessa luoduista näkymäelementeistä valitaan nimet ja kuvaukset sisältävät alueet ja sidotaan ne mallielementtien vastaaviin ominaisuuksiin.

Viittaavista näkymäelementeistä pitää pystyä navigoimaan mallielementin ensisijaiseen näkymäelementtiin. Ensisijaisen näkymäelementin näkymä avataan tarvittaessa, ja mikäli näkymä on tehty toisella työkalulla, työkalu käynnistetään ensin. Lopuksi ensisijainen näkymäelementti valitaan.

Muuttuneiden elementtien korostuksia pitää pystyä kytkemään päälle ja pois. Erilaisia korostuksia ovat mallinnussession aikana, baseline-kopioinnin jälkeen ja valitun aikavälin aikana tapahtuneiden muutosten korostaminen.

4.2.3 Hajautetun mallinnusympäristön vaatimukset

Trinity on hajautettu mallinnusympäristö, jossa useat käyttäjät voivat käsitellä samoja tietoja yhtä aikaa. Tämä asettaa lisävaatimuksia Trinityyn integroitaville mallinnustyökaluille, mitkä täytyy ottaa huomioon. Taulukkolaskentaohjelman laajennoksen tulee toimia välikomponenttina alkuperäisen ohjelman ja Trinityn tietokannan välillä. Sen tehtävänä on reagoida käyttäjien käyttöliittymässä tekemiin toimintoihin ja tallettaa muutokset tietokantaan sekä pitää näkymien tilat synkronoituna tietokannan kanssa.

Trinityn tietokantapohjaisuus ja tietokantakomponentti helpottavat rinnakkaisuuden hallintaa eikä työkalujen täydy erikseen hallita tietojen lukitsemista. Tietokannan tilaa muuttavat toiminnot tehdään transaktioissa, jotka suoritetaan pyyntöjen suoritusjärjestyksessä. Täten tietokannan tila säilyy eheänä myös yhtä aikaa tapahtuvissa toiminnoissa, ja viimeiseksi suoritettu toiminto jää voimaan.

Taulukko 4.4 listaa hajautetun mallinnusympäristön vaatimukset. Työkalujen tehtäväksi jää tietojen tallettaminen tietokantaan välittömästä muutosten havaitsemisen jälkeen sekä näkymien päivittäminen, kun tietokantakomponentti ilmoittaa muuttuneista tiedoista. Mallielementtien sekä näkymien muutokset saadaan helposti päivitettyä näkymiin melkein reaaliajassa. Työkalun pitää kuitenkin indikoida käyttäjälle jotenkin muiden käyttäjien tekemistä muutoksista, jotta käyttäjät eivät joudu hämmästelemään itsestään muuttuvia tietoja.

Taulukko 4.4. Hajautetun mallinnusympäristön vaatimukset.

Tunniste ja nimi	**Kuvaus**
V21. Mallielementtien muutosten päivittäminen	Mallielementin tietojen muuttumisen pitää päivittyä välittömästi näkymään riippumatta siitä, mistä työkalusta tai kuka muutoksen tekee.
V22. Näkymän muutosten päivittäminen	Näkymään tehtyjen muutosten pitää päivittyä näkymään välittömästi, kun toinen käyttäjä tekee muutoksen samaan näkymään.
V23. Muiden käyttäjien tekemien muutosten korostaminen	Muiden käyttäjien tekemät muutokset pitää indikoida jollain tapaa, jotteivät käyttäjät ihmettele itsestään muuttuvia tietoja.

4.2.4 Työkalun laajentamisvaatimukset

Työkalussa pitää varautua muuttuviin raportointi- ja mallinnuskäyttötarpeisiin sekä Trinityyn lisättäviin uusiin mallinnuskieliin.

Taulukko 4.5 listaa työkalun laajentamisvaatimukset. Toteutuksen pitää olla riittävän geneerinen, jotta koodiin ei tarvita muutoksia uusien näkymätyyppien tekemiseen. Näkymätyyppeihin on lisäksi pystyttävä määrittämään uusia template-elementtejä sekä muokkaamaan vanhoja. Näkymätyyppien ja template-elementtien ei tarvitse olla peruskäyttäjien tehtävissä, mutta niiden tekeminen täytyy olla riittävän yksinkertaista edistyneiden käyttäjien tehtäväksi.

Taulukko 4.5. Laajentamisvaatimukset.

Tunniste ja nimi	**Kuvaus**
V24. Uusien näkymätyyppien ja raporttipohjien lisääminen ilman koodiin tehtäviä muutoksia	Työkaluun pitää pystyä lisäämään uusia näkymätyyppejä ja raporttipohjia ilman, että työkalun koodiin joutuu tekemään muutoksia.
V25. Template-elementtien lisäys ja muokkaus	Näkymätyyppeihin pitää pystyä lisäämään uusia template-elementtejä sekä määrittämään niiden ominaisuudet ja ulkoasut. Lisäksi valmiita template-elementtejä on pystyttävä muokkaamaan.

5. EXCEL-LAAJENNUS TRINITYYN

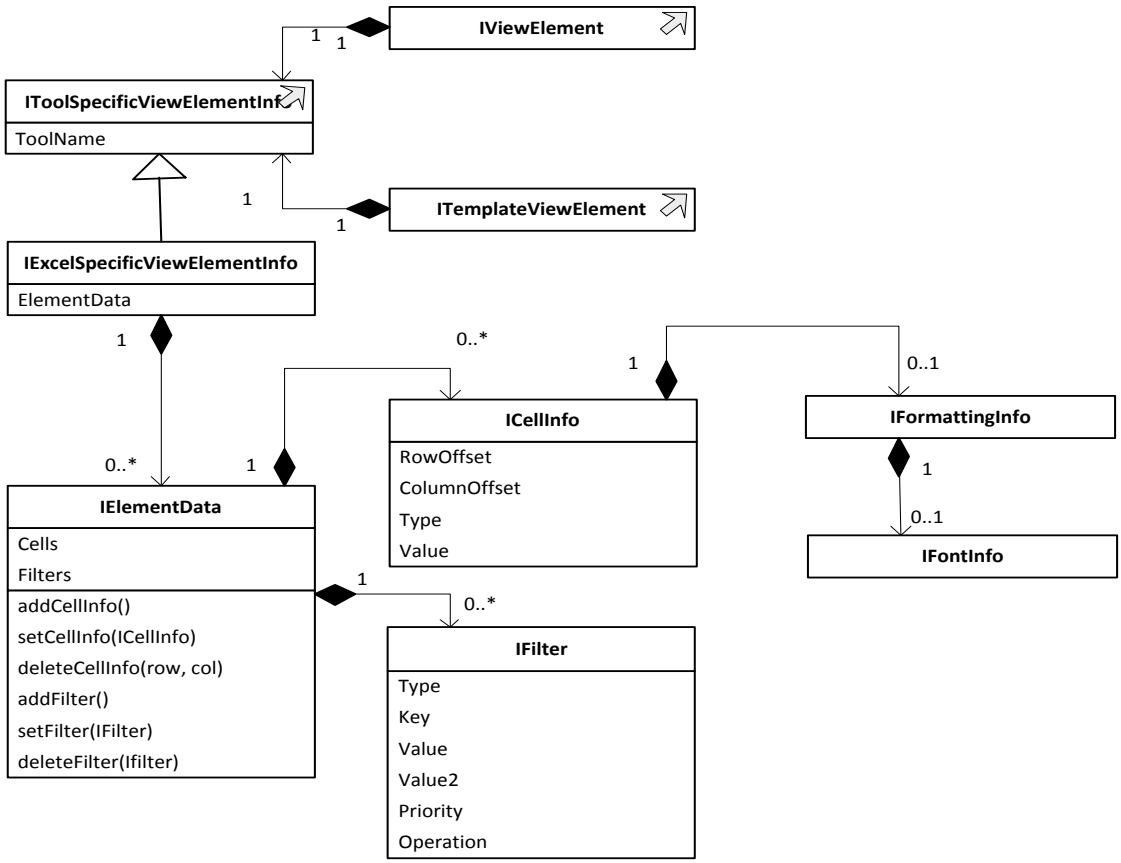
Trinityyn integroitavaksi taulukkolaskentaohjelmaksi valittiin Microsoft Excel sen yleisyyden vuoksi. Tässä luvussa kuvataan ratkaisut, joiden avulla se saadaan laajennettua luvun neljä vaatimukset toteuttavaksi raportointi- ja mallinnustyökaluksi. Luvussa esitellään toteutetun Trinityn Excel-laajennuksen tietosisältö sekä kuvataan toteutettu mallinnuskäyttöliittymä ja toimintojen sijoittuminen sen eri osiin. Lopuksi kuvataan työkalun käyttö sekä keskeisimmät toiminnot.

5.1 Tietosisältö

Trinityn metametamalli määrittää pitkälti Excel-laajennuksen tietosisällön rakenteen. Työkalun tietosisältö käyttää pohjana Trinityn näkymäelementtityyppiä ja siihen liittyvää työkalukohtaista tietoa sisältävää elementtiä. Ympäristöön lisätään uusi Excel-näkymäelementtityyppi, johon liittyy ominaisuuksia elementtien näyttämiseksi Excelin laskentataulukoissa, sekä Excel-näkymäelementteihin liittyvät työkalukohtaiset tiedot.

5.1.1 Excel-näkymäelementit

Excel-näkymäelementti muodostuu suorakaiteen muotoisesta solualueesta, josta tallennetaan solualueen vasemmassa yläkulmassa sijaitsevan solun rivi- ja sarakenumero sekä alueen korkeus ja leveys. Solualue rajoitetaan suorakaiteen muotoiseksi, koska Excel ei salli tiettyjen toimintojen, kuten kopioinnin, suorittamista epäyhtenäiselle solualueelle. Lapsielementtien solualueet kuuluvat isäelementin alueeseen. Isäelementin solualuetta kasvatetaan tarpeen vaatiessa, jos lisätyt lapsielementit eivät mahdu sen alueelle.
Kuva 5.1 esittää Excel-näkymäelementtiin liittyvät tiedot. Jokaiseen Excel-näkymäelementtiin kuuluu Excel-kohtaiset tiedot sisältävä elementti (IElementData). Työkalukohtaista tietoa ovat yksittäisistä elementin soluista talletettavat tiedot (ICell Info) sekä lista-elementeille asetettavat suodattimet (IFilter). Yksittäisen solun tietoihin liittyy lisäksi sen muotoilut (IFormattingInfo) sekä solun sisältämän arvon muotoilut (IFontInfo). Elementin solualueelle kuuluvista soluista tallennetaan kaikkien niiden solujen tiedot, joilla on jokin arvo tai niiden muotoiluja on muutettu. Solun sijainti tallennetaan poikkeamana näkymäelementin solualueen vasemman yläkulman solusta.



Kuva 5.1. Excel-näkymäelementtiin liittyvä työkalukohtainen tieto.

Taulukko 5.1: Erilaiset solutyypit.

Kenttä	**Selite**
Text	Solun arvo käsitellään tekstimuodossa.
Property	Solu on sidottu mallielementin ominaisuuteen. Tällöin solun Value kentässä oleva arvo kertoo ominaisuuden nimen.
Referen-ceIndicator	Kyseisessä solussa näytetään indikaattori, joka kertoo elementin olevan viittaava elementti.
FirstChild	Listaelementin template-elementillä oleva solutyyppi. Kertoo ensimmäisen lapsielementin sijainnin.
NextChild	Listaelementillä oleva solutyyppi, joka kertoo seuraavan lapsielementin sijainnin.
RowHeight	Laskentataulukkoelementin solutyyppi. RowOffset-kenttä kertoo, mistä rivistä on kyse ja Value-kenttään talletetaan kyseisen rivin korkeus.
Column-Width	Laskentataulukkoelementin solutyyppi. ColumnOffset-kenttä kertoo, mistä sarakkeesta on kyse ja Value-kenttään talletetaan kyseisen sarakkeen leveys.

Taulukko 5.1 esittää eri solutyyppien vaihtoehdot. Text-tyyppisistä soluista tallennetaan niiden arvo tekstimuodossa. Mallielementin ominaisuuteen sidotun solun tyyppi on Property. Muut solutyypit ovat tarkoitettu erityistarkoituksiin tietyn tyyppisillä elementeillä. Solun value-kenttään tallennetaan solun arvo, joka on Property-tyyppisen solun tapauksessa mallielementin ominaisuuden nimi.

CellFormatting ja Font-formatting sisältävät suuren joukon muotoiluun liittyviä ominaisuuksia. Niiden arvot asetetaan Excelin oliomallin Range- ja Font-olioilta saatavilta ominaisuuksilta.

5.1.2 Näkymätyypit ja template-elementit

Excel mallinnustyökaluun voidaan määritellä uusia näkymätyyppejä Trinityn tarjoamien laajennusmekanismien avulla. Näkymätyypit sisältävät Trinityn tarjoaman yleisen näkymätyypin ominaisuudet eli niihin kuuluvat template-elementit sekä näkymätyypin nimen. Lisäksi näkymätyypeillä on ominaisuus, joka määrittää onko näkymätyypin mukainen näkymä raportti vai muokattava näkymä.

Excel-template-elementeille voidaan asettaa metaominaisuuksia. Metaominaisuuksia käytetään määrittämään mallielementin tyyppi sekä suhde, joilla mallielementtejä kytketään kiinni toisiinsa.

Erilaisia template-elementtityyppejä ovat laskentataulukkoa edustava elementti, mallielementtiin liittyvä elementti, listaelementti, suhde-elementti (connector) sekä suhteen pää -elementti (connector end). Template-elementti voi esimerkiksi liittyä UML-luokkaan. Sille voidaan määritellä lapsielementeiksi luokan attribuutit ja operaatiot sisältävät listaelementit. Listaelementti-template sisältää metaominaisuuksina tiedon sen hyväksymästä lapsielementtityypistä, isäelementin tyypin sekä mallinnuskielessä määritetyn suhteen (relationship) sekä siihen liittyvien suhteenpäiden (relationship end) nimet, joita käytetään isä- ja lapsielementtien kytkemiseksi toisiinsa.

Suhde-elementti sisältää metaominaisuuksina siihen liittyvien suhteenpäiden tyypit tai mallielementtien tyypit, jos suhde kiinnitetään suoraan elementteihin. Lisäksi se sisältää mallinnuskielessä määritellyn suhteen ja suhteenpäiden nimet, joita käytetään mallielementtien kytkemiseen toisiinsa samaan tapaan kuin listaelementtien tapauksessa. Suhde-elementtejä ovat esimerkiksi UML-periytyminen, joka kiinnitetään suoraan kahden luokan välille sekä UML-assosiaatio, joka puolestaan kiinnitetään suhteen pää -elementteihin, jotka kiinnitetään luokkiin.

Suhteen pää -elementti sisältää metaominaisuuksina tiedon siihen kiinnitettävästä elementtityypistä sekä mallielementtien kiinnittämiseen tarvittavat tiedot. Esimerkiksi UML-assosiaatiolla olevat UML-propertyt ovat suhteen päitä, joiden ominaisuuksia ovat muun muassa lukumääräsuhde, nimi ja navigoitavuus. Ne kiinnitetään UML-assosiaation ja UML-luokan välille.

5.1.3 Listaelementteihin liittyvät suodattimet

Sekä ylimmän tason listaelementeille että lapsielementteinä jollakin näkymäelementillä oleville listaelementeille voidaan asettaa suodattimia. Taulukko 5.2 listaa suodattimista talletettavien tietojen tietosisällön. Suodattimilla on tyyppi, joka määrittää avain- ja arvokenttien käyttötarkoituksen. Prioriteetti määrittää, missä järjestyksessä suodattimet käydään läpi ja elementtien suodatus tapahtuu. Operaattori-kenttään talletetaan looginen operaattori, kuten AND tai OR, joka määrittää miten samalla prioriteetilla olevat suodattimet suhtautuvat tosiinsa. Suodattimia voidaan siis asettaa esimerkiksi siten, että kahden suodattimen asettamien ehtojen on molempien toteuduttava tai riittää, että vain toinen toteutuu.

Taulukko 5.2. Suodattimista talletettavat tiedot.

Kenttä	**Selite**
Type	Suodattimen tyyppi
Key	Suodattimen avain. Merkitys riippuu suodattimen tyypistä.
Value	Suodattimen arvo, jonka perusteella elementtejä suodatetaan.
Value2	Toinen suodattimen arvo, jota tarvitaan tietyn tyyppisille suodattimille.
Priority	Prioriteetti määrittää, missä järjestyksessä suodattimet käydään läpi ja elementit suodatetaan.
Operator	Looginen operaattori saman prioriteetin suodattimien yhdistämiseksi.

Taulukko 5.3 esittää erilaiset suodatintyypit ja suodattimen avaimen sekä arvojen merkityksen kyseisellä suodatintyypillä. ”Property equals”-tyyppinen suodatin suodattaa pois kaikki mallielementit, joiden ominaisuuden arvo ei ole täsmälleen annettu arvo. Avain kentässä kerrotaan mallielementin ominaisuuden nimi ja arvo kentässä sen arvo.

”Property greater”- ja ”Property smaller”-tyyppiset suodattimet jättävät suodattamatta vastaavalla tavalla mallielementit, joiden arvo on suurempi tai pienempi kuin suodattimeen asetettu arvo. ”Property between”-tyyppinen suodatin puolestaan määrittää ominaisuuden arvon ala- ja ylärajan, joiden välillä sen on oltava. Näitä suodatintyyppejä voidaan käyttää vain luku- ja aikatyyppisten ominaisuuksien kanssa.

”Property begins”- ja ”Property contains”-tyyppisiä suodattimia käytetään suodattamaan mallielementtejä merkkijonotyyppisten ominaisuuksien arvojen perusteella. Ensin mainittu vaatii, että ominaisuuden arvo alkaa annetulla merkkijonolla ja jälkimmäiselle riittää, että annettu merkkijono löytyy jostain kohti ominaisuuden arvoa.

”Connected elements”-tyyppisten suodattimien arvona voi olla joko ”Exists” tai ”Does not exist”. Ensimmäisessä tapauksessa elementtiin pitää liittyä vähintään yksi määritellynlainen mallielementti ja jälkimmäisessä niitä ei saa liittyä yhtään. Suodattimen arvo määrittää liittyvän mallielementin tyypin ja toinen arvo määrittää suhteen nimen, jolla elementit on kytketty kiinni toisiinsa.

”Has tag”-tyyppinen suodatin suodattaa kaikki mallielementit, joille ei ole asetettu suodattimeen määritettyä avainsanaa. Tässä tapauksessa avain-kenttä sisältää avainsanan tyypin ja arvo-kenttä itse avainsanan.

Taulukko 5.3. Erilaiset suodatintyypit.

Tyyppi	**Avain**	**Arvo**	**Arvo2**	**Selite**
”Property equals”	Ominaisuuden nimi	Ominaisuuden arvo	–	Ominaisuuden arvo täsmälleen.
”Property greater”	Ominaisuuden nimi	Ominaisuuden arvo	–	Ominaisuuden arvo suurempi.
”Property smaller”	Ominaisuuden nimi	Ominaisuuden arvo	–	Ominaisuuden arvo pienempi.
”Property between”	Ominaisuuden nimi	Ominaisuuden arvon minimi	Ominaisuuden arvon maksimi	Ominaisuuden arvo annettujen välissä.
”Property begins”	Ominaisuuden nimi	Ominaisuuden arvo	–	Ominaisuuden arvo alkaa.
”Property contains”	Ominaisuuden nimi	Ominaisuuden arvo	–	Ominaisuuden arvo sisältää.
”Connected Elements”	”Exists”	Mallielementtityypin nimi	Suhteen nimi	Mallielementtiin liittyy vähintään yksi määritetty elementti.
”Connected Elements”	”Does not exist”	Mallielementtityypin nimi	Suhteen nimi	Mallielementtiin ei liity yhtään määritetty elementti.
”Has Tag”	Avainsanan tyyppi	Avainsana	–	Elementille on asetettu määrätty avainsana.

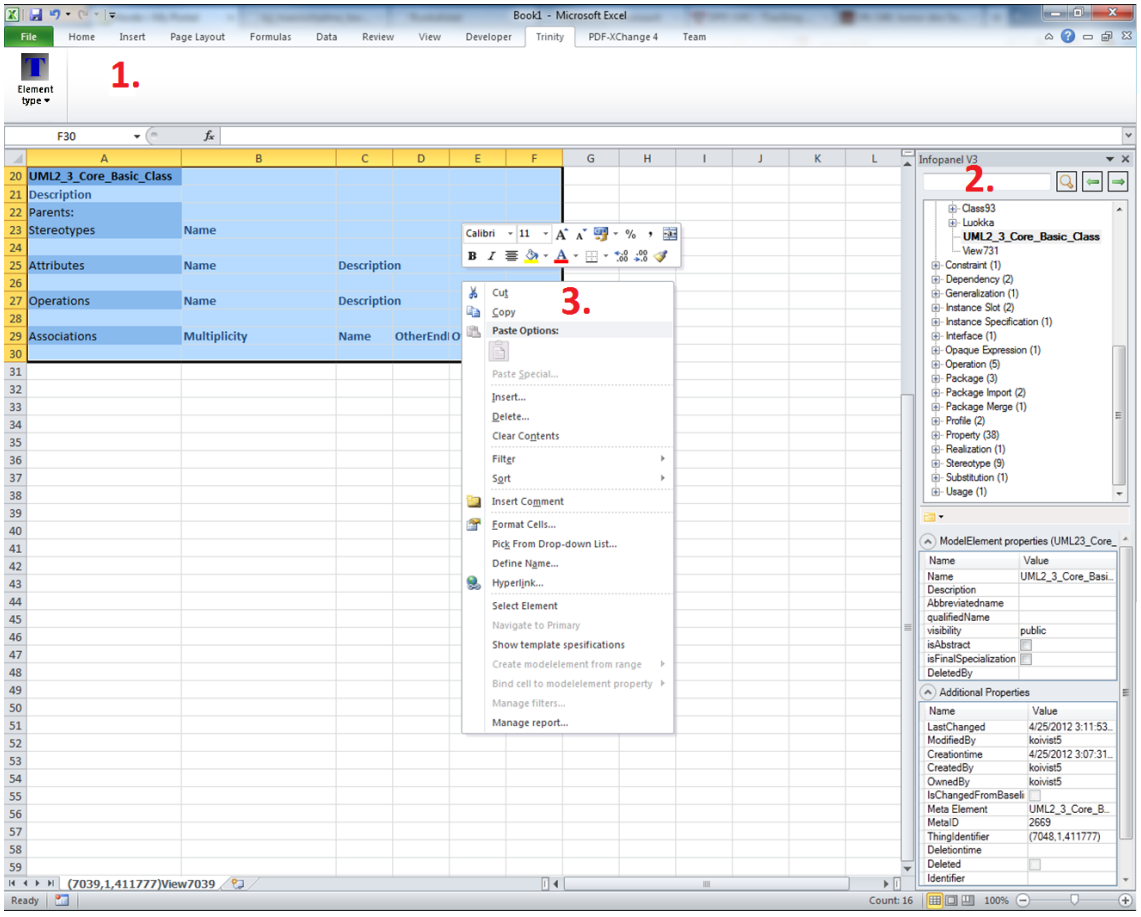
5.2 Laajennuksen mallinnuskäyttöliittymä

Mallinnustyökalun käyttöliittymässä pyritään säilyttämään Excelin alkuperäinen ulkoasu, jotta työkalun käyttäminen olisi mahdollisimman tuttua Exceliä aiemmin käyttäneille henkilöille. Excelin omia toimintoja ei rajoiteta, mikäli se on mahdollista, eikä käyttäjälle näytetä turhia dialogeja tai valikoita. Nämä vastaavat vaatimusta V04.

Mallinnustoiminnot, joita ei voi tehdä Excelin omilla toiminnoilla, sijoitetaan mallinnuskäyttöliittymään (Kuva 5.2). Laajennuksen mallinnuskäyttöliittymä sisältää kolme paikkaa mallinnustoimintojen sijoittamiseksi:

1. Excelin valintanauhassa [31] sijaitseva Trinity-välilehti.
2. Trinityn Visio-laajennuksesta tuttu sivupalkki.
3. Valittuun solualueeseen liittyvä kontekstivalikko.

Excelin valintanauhan Trinity-välilehti tulee näkyviin, kun työkaluun avataan näkymä käsiteltäväksi. Välilehti sisältää ponnahdusvalikon, josta näkymään voidaan luoda uusia elementtejä vaatimuksen V09 mukaisesti. Ponnahdusvalikko sisältää kaikki mallielementtityypit, joille on määritelty template-elementti avoimen näkymän näkymätyyppiin.



Kuva 5.2. Excel-laajennuksen mallinnuskäyttöliittymä.

Excelin oliomalli tarjoaa mahdollisuuden kustomoitavien paneelien (CustomTaskPane) [32] luomiseen ajonaikaisesti. Paneeleihin voidaan lisätä käyttöliittymiä, jotka toteuttavat UserControl-luokan. Paneelit ovat siirrettävissä ja niiden kokoa voidaan muuttaa. Lisäksi ne voidaan ankkuroida työkirjan laitoihin ja ylä- sekä alareunoihin.

Mallinnustyökalu sisältää paneelin, johon on sijoitettu Trinityn informaatiopaneeli. Paneeli on näkyvissä niissä laskentataulukoissa, joissa on näkymä avattuna. Se ankkuroidaan oletuksena työkirjan oikeaan reunaan. Informaatiopaneeli tarjoaa suoraan osan mallinnustoimintojen vaatimuksista:

- Valittujen elementtien ominaisuuksia voidaan muokata elementin informaatioikkunasta (V07).
- Listaelementin ollessa valittuna sille voidaan luoda uusia lapsielementtejä sekä niitä voidaan poistaa (V14 ja V15).
- Mallielementtipuusta voidaan raahata ja pudottaa elementtejä näkymään, jolloin niille luodaan kyseiseen kohtaan viittaava näkymäelementti (V10).
- Mallielementtipuusta voidaan navigoida ensisijaiseen näkymäelementtiin sekä elementin informaatioikkunasta voidaan navigoida kaikkiin mallielementin näkymäelementteihin (V19).

Käyttäjän painaessa hiiren kakkospainiketta jonkin solualueen ollessa valittuna avautuu soluihin liittyvä kontekstivalikko. Tässä kontekstivalikossa näytetään mallinnustoiminnot, jotka ovat kytkettynä päälle tai pois sen mukaan, mitä valitulla solualueella sijaitsee.

Kun yhden elementin solualueelta on valittuna yksi tai useampi solu, on elementin koko solualueen valitseminen kytkettynä päälle. Jos elementin solualueelta on valittuna vain yksi solu, voidaan kyseisen solun arvo sitoa mallielementin ominaisuuteen. Jos valittu elementti on viittaava näkymäelementti, navigointi ensisijaiseen näkymäelementtiin on kytkettynä päälle (V19). Jos valittu solualue ei kuulu millekään elementille, päälle kytkettynä toimintona on ainoastaan uuden elementin luominen kyseisestä solualueesta.

5.3 Käyttö ja toiminnot

Laajennuksessa pyritään säilyttämään Excelin perustoiminnallisuus, ja mahdollisimman suuri osa mallinnustoiminnoista tehdään Excelin omilla toiminnoilla. Esimerkiksi mallielementtien ominaisuuksien arvoja voidaan muokata kirjoittamalla ominaisuuden arvon sisältävään soluun uusi arvo (V07).

Osa mallinnustoimintojen määritellyistä käyttäytymisistä tulee Trinityn ympäristön toisista mallinnustyökaluista. Toimintojen on oltava yhdenmukaisia kaikissa Trinityn työkaluissa, jotta kaikkien työkalujen käyttö olisi mahdollisimman samanlaista ja loogista. Esimerkiksi elementtien kopioinnin on toimittava Excel-laajennuksessa saman lailla kuin Trinityn Visio-laajennuksessa.

Kaikkien toimintojen kumoaminen ja uudelleen tekeminen tapahtuu Excelin näihin tarkoitetuilla painikkeilla tai vaihtoehtoisesti pikanäppäimillä (V06). Toimintojen kumoaminen palauttaa tietokannan sekä laskentataulukon tilaan, jossa ne olivat ennen toimintojen suorittamista.

5.3.1 Raportointi

Excel-raportit, kuten muutkin näkymät, luodaan ja avataan Trinityn hallintakäyttöliittymästä. Raportin ensimmäisellä avauskerralla näytetään dialogi raportin asetuksille. Dialogiin raahataan ja pudotetaan hiirellä raportin lähteeksi asetettavat mallit tai näkymät hallintakäyttöliittymästä. (V01)

Dialogista pääsee lisäksi hallitsemaan listaelementteihin kohdistuvia suodattimia. Käyttäjä voi valita vetovalikosta minkä tahansa raporttipohjaan kuuluvan listaelementti-templaten, jonka suodattimia haluaa muokata. Suodattimet kohdistuvat kaikkiin raportin listaelementteihin, joille asetetaan kyseisiä template-elementtejä. Hän voi luoda dialogin avulla uusia suodattimia sekä poistaa ja muokata asetettuja suodattimia (V02).

Jos käyttäjä haluaa asettaa edellisessä luvussa (4.2.1) kuvatussa esimerkkitapaus 1:ssä mainitut suodattimet raporttiin, valitsee hän ensiksi luokat sisältävän ylimmän tason listaelementti-templaten. Seuraavaksi hän lisää sille ”Property equals”-tyyppisen suodattimen, jonka avaimeksi hän asettaa UML-luokan createdBy-ominaisuuden ja arvoksi oman käyttäjänimensä. Tällä suodattimella raporttiin jää vain käyttäjän itsensä tekemät luokat. Lisäksi hän lisää ”Property greater”-tyyppisen suodattimen, jonka avaimeksi hän asettaa creationTime-ominaisuuden ja arvoksi kuukausi sitten olleen ajanhetken. Tämä suodatin suodattaa pois kaikki luokat, jotka on luotu yli kuukausi sitten.

Seuraavaksi käyttäjä valitsee luokalle kuuluvan operaatiot sisältävän listaelementti-templaten, jolle hän lisää Property equals -tyyppisen suodattimen, asettaa sille avaimeksi UML-operaation visibility-ominaisuuden sekä arvoksi public. Tämän suodattimen avulla luokkiin tulee mukaan vain ne operaatiot, joiden näkyvyysmääre on public. Lisäksi hän lisää ”Connected elements”-tyyppisen suodattimen, jonka avaimeksi hän asettaa ”Does not exist”, ensimmäiseksi arvoksi UML-property ja toiseksi avaimeksi UML:n mallinnuskieleen määritetyn suhteen nimen, jolla parametrit kiinnitetään operaatioihin. Tämän suodattimen avulla suodatetaan pois kaikki operaatiot, joihin liittyy vähintään yksi parametri.

Raportin asetuksia on mahdollista muokata myös raportin generoinnin jälkeen saman dialogin kautta valitsemalla raportin asetukset valintanauhasta. Myös suodattimia voi asettaa jälkeenpäin joko template-elementtikohtaisesti tai yksittäisille listaelementeille saman dialogin avulla. Jos suodattimia muutetaan template-elementtikohtaisesti, asetetut suodattimet lisätään template-elementin mukaisiin lista-elementteihin automaattisesti niiden luomisen yhteydessä. Raportit päivitetään vastaamaan muuttunutta tilanne, kun uudet asetukset tai suodattimien muutokset hyväksytään.

Muualla tehdyt muutokset mallielementtien ominaisuuksiin ja suhteisiin päivittyvät automaattisesti generoituihin raportteihin – kuten muihinkin Excel-näkymiin. Tämän lisäksi raportteihin lisätään näkymäelementit lähdemalleihin ja näkymiin luoduille elementeille, mikäli ne täyttävät asetettujen suodattimien ehdot. Lisäys tehdään, kun näkymä avataan uudelleen. Raporteista poistetaan näkymäelementit, jos niihin viittaavi-

en mallielementtien ominaisuuksia muutetaan siten, että ne eivät enää täytä suodatusehtoja. Nämä päivitystoiminnot vastaavat vaatimusta V03.

5.3.2 Näkymien ulkoasun muokkaaminen

Tekstin kirjoittaminen mihin tahansa laskentataulukon soluun ja solujen sekä tekstin muotoilujen muutokset tallennetaan välittömästi tietokantaan. Jos muutoksia tehdään jonkin elementin solualueelle, talletetaan muutokset kyseisen näkymäelementin tietoihin. Jos taas muutetut solut eivät kuulu millekään elementille, tiedot talletetaan laskentataulukkoa edustavan näkemäelementin tietoihin. Niille tallennetaan myös rivien ja sarakkeiden koot sekä ryhmittelyt. (V08)

Näkymäelementtien sijainteja saadaan muutettua valitsemalla elementin solualue ja suorittamalla Excelin leikkaa ja liitä -toiminto. Valitun solualueen voi myös siirtää painamalla hiiren ykköspainike pohjaan solualueen reunassa, raahaamalla solualue toiseen kohtaan laskentataulukossa ja päästämällä painikkeesta irti. Näkymäelementtien sijainnit muuttuvat myös silloin, kun laskentataulukkoon lisätään uusia sarakkeita tai rivejä ja kun niitä poistetaan.

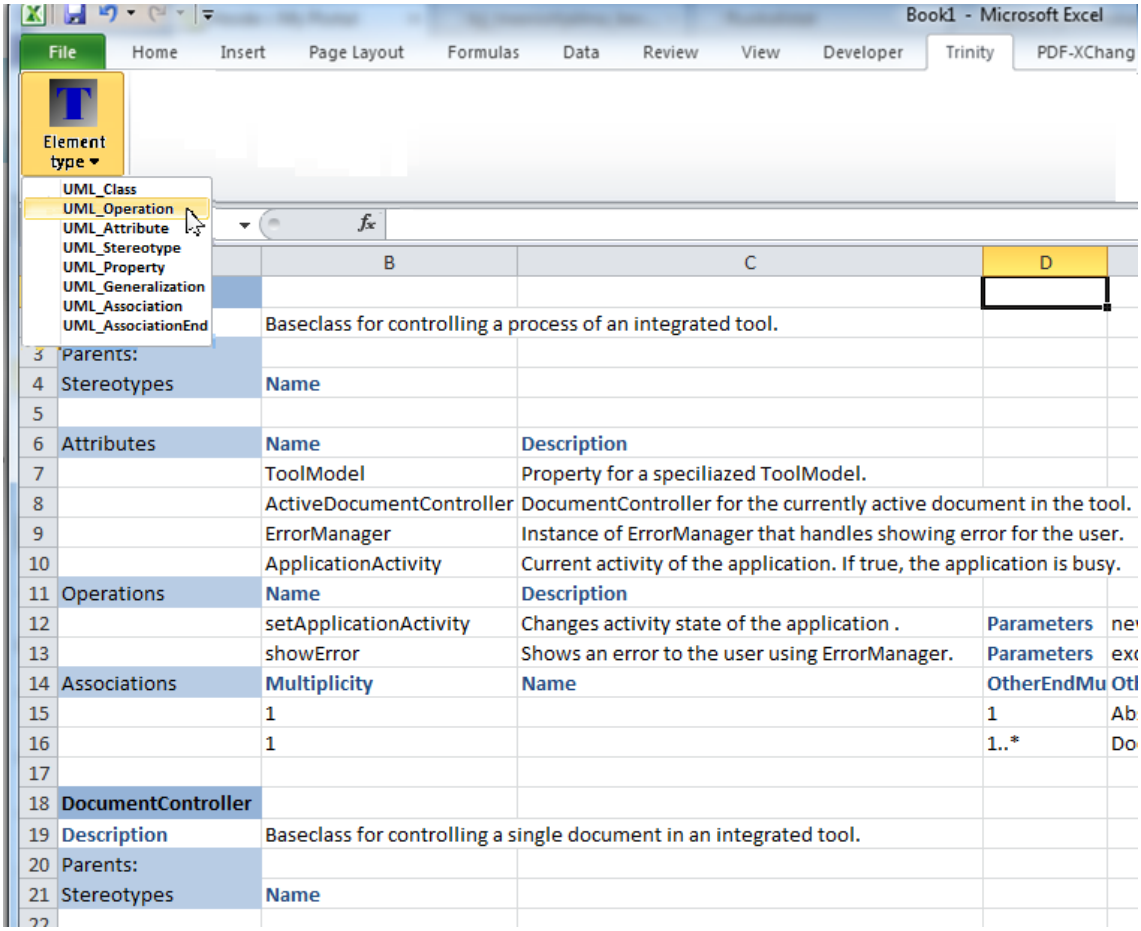
Jos lapsielementti siirretään pois isäelementin alueelta, puretaan niiden väliltä isä-lapsi-suhde. Mikäli elementti siirretään toisen elementin solualueelle, joka hyväksyy siirrettävän elementin lapseksi, siirretään lapsielementti automaattisesti oikeaan kohtaan isäelementin alueella ja luodaan elementtien välille isä-lapsi-suhde (V16).

5.3.3 Elementtien luominen ja poistaminen

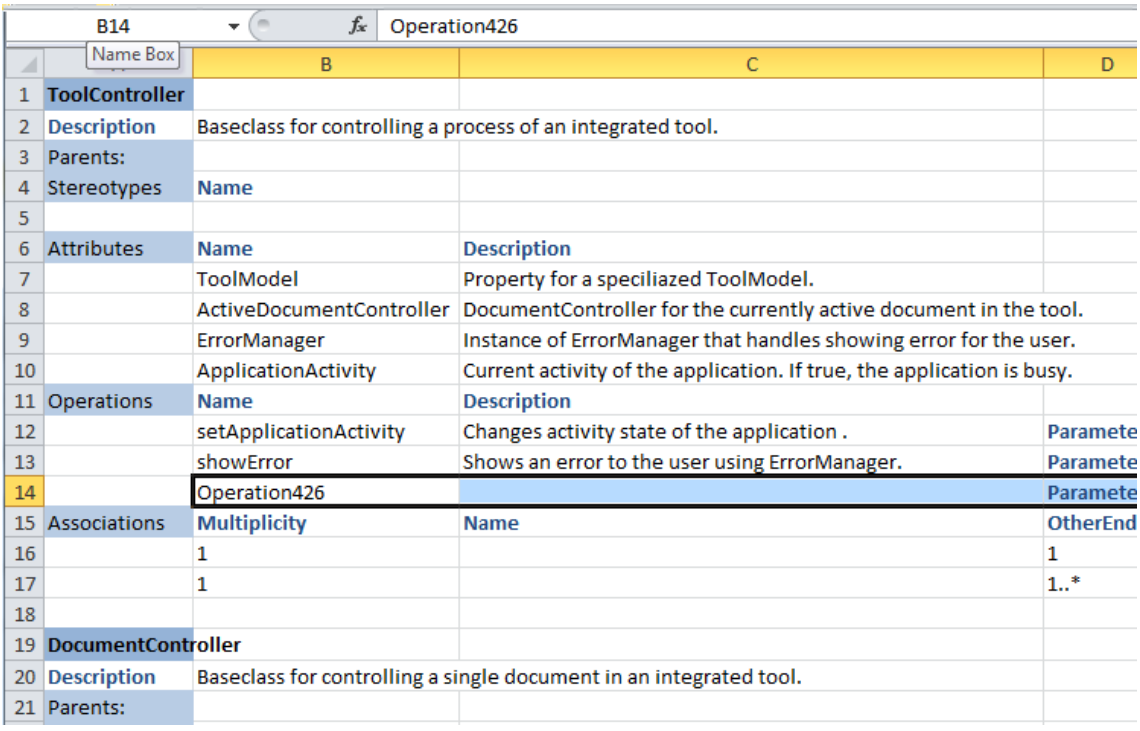
Excel-näkymien kautta pystyy luomaan uusia mallielementtejä valitsemalla laskentataulukosta kohdan, johon haluaa elementin lisättävän ja suorittamalla lisää elementti -toiminnon valintanauhasta sekä valitsemalla template-elementin luotavalle elementille (V09). Elementille luodaan näkymään ensisijainen näkymäelementti, jonka ulkoasu asetetaan valitulta template-elementiltä. Lisäksi näkymän malliin luodaan uusi mallielementti, jonka tyyppi on määritelty template-elementissä.

Elementin luominen lapsielementiksi tapahtuu valitsemalla elementti, joka hyväksyy luotavan elementin lapseksi ja suorittamalla lisää elementti -toiminto. Tämän seurauksena luodaan uusi elementti ja se asetetaan lapseksi valitun elementin oikeaan kohtaan (V14). Jos valittuna on elementti, jolle lisättävää elementtiä ei voida asettaa lapsielementiksi, siirretään luotavaa elementtiä alaspäin seuraavaan vapaaseen kohtaan laskentataulukossa.

Kuva 5.3 esittää tilannetta, missä luokalta on valittuna yksittäinen solu ja sille ollaan lisäämässä uutta UML-operaatioita lapsielementiksi. Luokka ei hyväksy operaatiota lapseksi, mutta sillä lapsena oleva operaatiolista hyväksyy. Kuva 5.4 esittää lisäyksen jälkeisen tilanteen. Uusi operaatio lisättiin viimeiseksi elementiksi luokan operaatiolistaan ja uuden operaation koko solualue valittiin. Operaatiolle lisättiin uusi rivi laskentataulukkoon, joten sen alapuolella olevia elementtejä siirrettiin alaspäin.



Kuva 5.3. UML-operaation lisääminen luokalle lapseksi.



Kuva 5.4. Tilanne operaation lisäyksen jälkeen.

Viittaavien näkymäelementtien luominen olemassa oleville mallielementeille onnistuu hallintakäyttöliittymästä ja minkä tahansa Trinityn työkalun informaatiopaneelin puunäkymästä (V10). Lisäys tapahtuu valitsemalla puurakenteesta yksi tai useampi mallielementti, raahaamalla se hiirellä haluttuun kohtaan laskentataulukkoa ja päästämällä napista irti. Elementin luominen käyttäytyy saman lailla kuin uutta mallielementtiä luotaessa, paitsi että tässä tapauksessa ei luoda uutta mallielementtiä, vaan viittaus olemassa olevaan mallielementtiin.

Näkymässä olevien elementtien poistaminen tapahtuu tyhjentämällä elementin koko solualueen sisältö tai poistamalla kaikki sille kuuluvat solut esimerkiksi poistamalla kokonaisia rivejä tai sarakkeita laskentataulukosta. Mikäli poistettavalla elementillä on lapsielementtejä, myös ne poistetaan, koska ne sijaitsevat poistettavan isäelementin solualueella. Nämä erilaiset poistotavat toteuttavat vaatimuksen V11. Lapsielementin poistaminen tapahtuu valitsemalla lapsielementin solualue ja poistamalla se laskentataulukosta (V15). Tällöin isäelementti säilytetään ja lapsielementti poistetaan.

5.3.4 Elementtien kopiointi

Trinityn tarjoamia kopiointitapoja ovat täysikopiointi, viittauskopiointi, normaalikopiointi ja siirtäminen, jotka Felin kuvaa diplomityössään [27]:

- Täysikopioinnissa sekä malli että näkymäelementti kopioidaan kohdenäkymään ja uusi näkymäelementti asetetaan viittaamaan uuteen mallielementtiin.
- Viittauskopioinnissa luodaan viittaava näkymäelementti, joka viittaa alkuperäiseen mallielementtiin.
- Normaalissa kopioinnissa suoritetaan täyskopiointi, mikäli alkuperäinen näkymäelementti on ensisijainen näkymäelementti, muussa tapauksessa viittauskopiointi.
- Siirtämisessä suoritetaan täyskopiointi ja poistetaan alkuperäinen elementti.

Elementit kopioidaan Excelin kopioi ja liitä -toiminnolla (V12). Käyttäjä valitsee elementin solualueen tai osan siitä ja kopioi sen leikepöydälle. Liittämisen voi tehdä työkalun samaan näkymään tai eri näkymään. Saman näkymän sisällä kopioitaessa kopiointitapana on täysikopiointi. Näkymien välillä kopioitaessa käyttäjälle näytetään dialogi, josta hän voi valita haluamansa kopiointitavan.

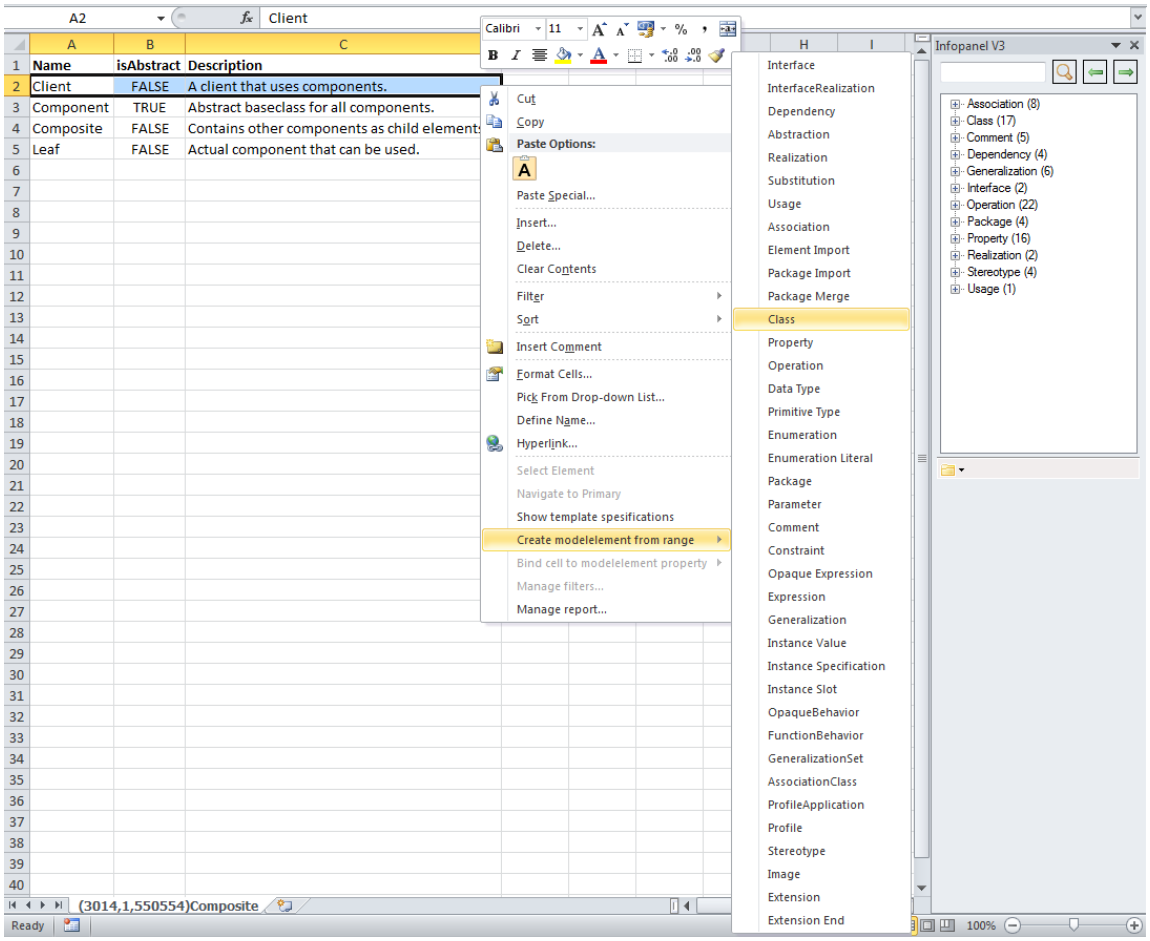
Elementtejä voi kopioida elementille lapseksi kopioimalla halutut elementit normaaliin tapaan ja liittämällä ne halutun isäelementin alueelle. Kopioidut lapsielementit siirretään oikeaan kohtaan ja lapsielementtejä varten lisätään tarvittaessa uusia rivejä kuten uutta lapsielementtiä lisätessä.

Elementtejä kopiointi onnistuu myös Trinityn toisiin työkaluihin ja niistä voidaan kopioida elementtejä Exceliin (V13). Kun Excelistä on kopioitu elementtejä leikepöydälle ja laskentataulukko poistuu aktiivisesta tilasta, vaihdetaan leikepöydän sisällöksi kopioituihin elementteihin viittaava Trinity-URI. Kaikki Trinityn työkalut ymmär-

tävät tämän URI-muodon ja osaavat luoda sen tiedoista elementeille näkymäelementit, kun URI:n sisältämä teksti liitetään työkalun näkymään.

5.3.5 Tietojen liittäminen osaksi mallia

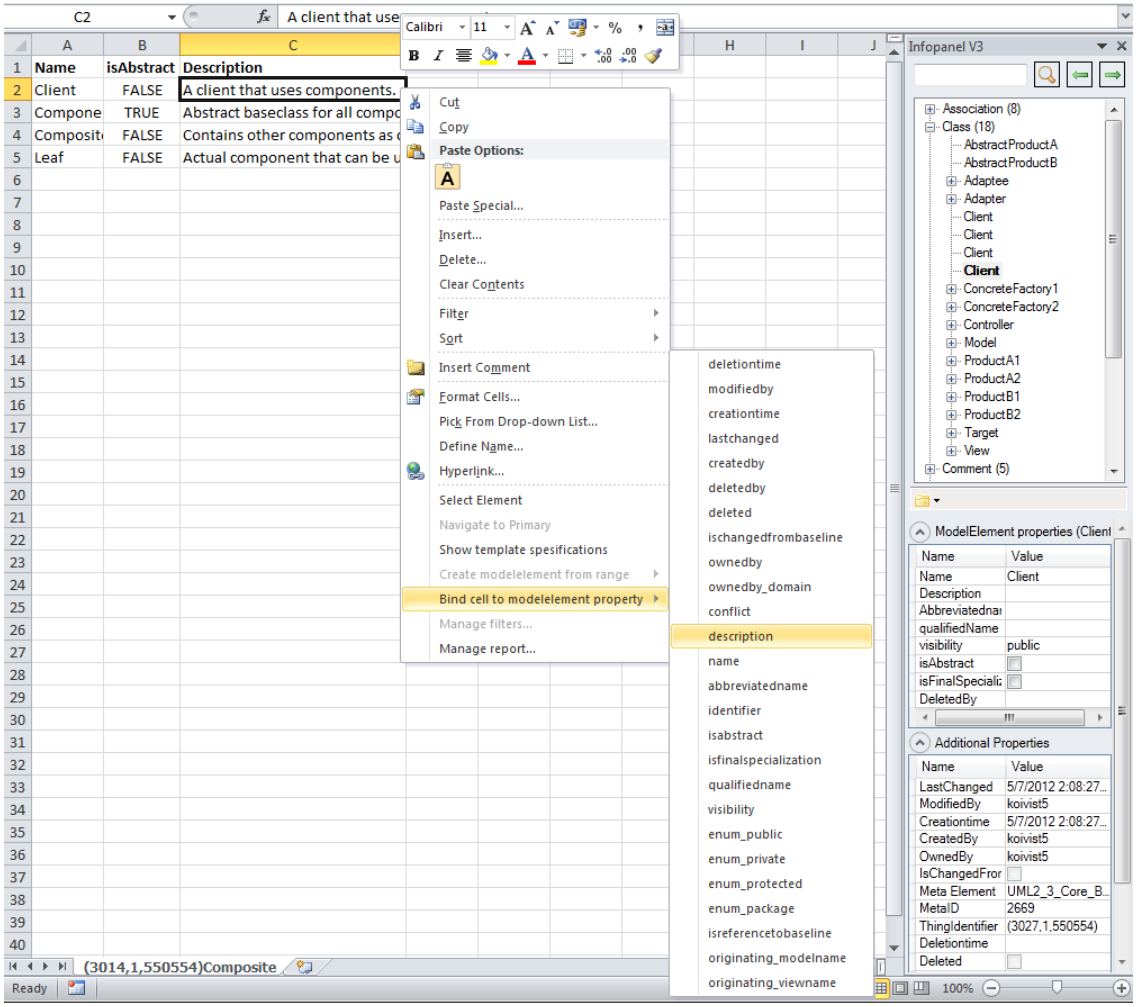
Excel-työkalun avulla on myös mahdollista liittää valmiina olevia tietoja osaksi malleja. Tämä tapahtuu luomalla uusi Excel-näkymä ja kopioimalla halutut tiedot siihen. Tiedot asetetaan laskentataulukolle tekstimuodossa. Tiedoista pystyy luomaan mallielementtejä valitsemalla haluttu solualue laskentataulukosta ja luomalla alueesta uusi valitun tyyppinen mallielementti kontekstivalikossa sijaitsevalla toiminnolla. Kuva 5.5 näyttää tästä esimerkin. Toiminnon suorittamisen jälkeen luodaan valitun tyyppinen mallielementti sekä sille ensisijaisen näkymäelementti. Näkymäelementti sisältää valittuun alueeseen kuuluvat solut, joiden sisältö tallennetaan elementille. Tämä toiminto vastaa vaatimusta V17.



Kuva 5.5. Laskentataulukossa olevan tiedon annotointi UML-luokaksi.

Näkymäelementeille kuuluvia tekstimuodossa talletettuja tietoja voi tämän jälkeen sitoa mallielementin ominaisuuksiin valitsemalla solu kontekstivalikossa sijaitsevalla toiminnolla ja päättämällä mihin ominaisuuteen se sidotaan. Kuva 5.6 näyttää tästä esimerkin. Mikäli sidotussa solussa on jokin arvo, asetetaan se mallielementin ominaisuuden uudeksi arvoksi. Jos solu on tyhjä, tulostetaan siihen ominaisuuden nykyinen

arvo. Sitomisen jälkeen ominaisuuteen tehdyt muutokset päivittyvät kyseiseen soluun. Tämä toiminto vastaa vaatimusta V18.



Kuva 5.6. Annotoidun luokan solussa olevan tiedon liittäminen luokan ominaisuuteen.

Käyttäjällä voi esimerkiksi olla valmiina elementtien nimiä ja ominaisuuksia työkirjassa taulukoituna, jotka hän haluaa saada siirrettyä osaksi mallia. Hän luo uuden Excel-näkymän malliin, johon haluaa lisätä elementit ja avaa kyseisen näkymän käsiteltäväksi Excel-laajennokseen. Tämän jälkeen hän kopioi tiedot valmiista työkirjasta näkymään, johon ne tallentuvat tekstimuotoisena. Kuva 5.5 esittää tilannetta, jossa käyttäjä valitsee luokan tiedot sisältävän solualueen laskentataulukosta ja annotoi sen luokaksi. Lopuksi hän vielä sitoo niiden nimet ja muut ominaisuudet sisältävät solut mallielementtien vastaaviin ominaisuuksiin. Kuva 5.6 esittää tilanteen, jossa sidotaan annotoidun luokan kuvauksen sisältämä solu sen luokan mallielementin kuvaukseksi. Näin irralliset tiedot saatiin osaksi mallia ja niitä voidaan käsitellä vaikka Trinityn toisella työkalulla.

5.3.6 Elementtien muutosten korostaminen

Työkalussa voidaan kytkeä päälle ja pois erilaisia elementtien muutosten korostamisia. Näitä ovat muun muassa mallinnussession aikana tapahtuneiden muutosten korostaminen, muutosten vertaaminen baseline-malliin ja valittuna aikavälinä muuttuneiden elementtien korostaminen. Erilaiset muutosindikoinnit on toteutettu lisäämällä elementtien solualueen ympärille reunaviiva ja värjäämällä se tietyillä väreillä eri tilanteissa. Muuttuneet elementit värjätään sinisellä ja uudet elementit vihreällä. (V20)

Muutosten korostamisia ollaan muuttamassa työn kirjoitushetkellä Trinityn Visio-mallinnustyökalussa ja yhtenäistämässä niitä koko ympäristössä, joten väritykset tulevat muuttumaan myös Excel-mallinnustyökalussa. Erilaisia muutosten korostamisia tullaan todennäköisesti ilmaisemaan edellä mainitun reunaviivojen värjäämisen lisäksi elementeissä käytetyn fontin värityksillä.

Muiden käyttäjien tekemät muutokset tullaan myös ilmaisemaan värityksillä (V23), kunhan muutoksissa käytetyt värit saadaan päätettyä. Jokaisesta muutoksesta tallennetaan elementin uusi versio Trinityn tietokantaan, mutta kyseisten tietojen tarkasteluun ei ole toteutettu käyttöliittymää. Käyttöliittymä tullaan todennäköisesti lisäämään työkalun sivupalkin välilehteen, josta on mahdollista tarkastella muutoshistoriaa.

5.3.7 Hajautetun mallinnusympäristön huomioiminen

Trinityn tietokantakomponentti tarjoaa mahdollisuuden kuunnella mallielementtien ominaisuuksien muutoksista laukaistavia tapahtumia. Excel-laajennus kuuntelee näitä tapahtumia kaikilta näkymässä olevilta elementeiltä ja päivittää solut, joita muutos koskee. Tämä vastaa vaatimusta V21.

Tietokantakomponentti tarjoaa myös mahdollisuuden kuunnella näkymäelementtien ominaisuuksien muutoksiin liittyviä tapahtumia. Työkalu kuuntelee myös näitä ja reagoi niihin välittömästi. Näkymäelementit siirretään oikeaan kohtaan näkymässä, ja niiden koot muutetaan vastaamaan tietokannan tilannetta, mikäli elementin rivi, sarake, leveys tai korkeus muuttuu. Mikäli näkymäelementtien solujen tiedot muuttuvat, solut ladataan tietokannasta ja näkymäelementit piirretään uudelleen. Näkymäelementtien isä-lapsi-suhteen vaihtuessa ne siirretään tietokantaan talletettuun kohtaan näkymässä. Nämä vastaavat vaatimusta V22.

5.3.8 Näkymätyyppien ja template-elementtien määrittely

Uusien näkymätyyppien lisäämiseksi Trinityn tietokantaan täytyy lisätä sille tarvittavat tiedot (V24). Peruskäyttäjien ei ole tarkoitus pystyä lisäämään uusia näkymätyyppejä. Kirjoitushetkellä näkymätyyppien lisäämiselle ei ole toteutettu käyttöliittymää, vaan lisäämisen on hoitanut Trinityn tietokantavastaava.

Excel-näkymän ollessa auki pystyy kyseisen näkymän näkymätyypin template-elementit avaamaan muokattavaksi. Tämä toiminto on kuitenkin piilotettu peruskäyttäjiltä. Näkymätyypin template-elementeille luodaan uusi laskentataulukko, johon niiden

määritellyt ulkoasut asetetaan. Solualueeseen liitettyyn konteksti-valikkoon lisätään toiminnot uusien template-elementtien lisäämiseksi ja valittujen muokkaamiseksi sekä poistamiseksi. Nämä toiminnot vastaavat vaatimusta V25.

Uusi template-elementti luodaan määrittelemällä ensin sen ulkoasu laskentataulukkoon ja valitsemalla haluttu solualue. Sen jälkeen valitaan kontekstivalikosta uuden template-elementin luonti sekä tyyppi luotavalle template-elementille. Tämän jälkeen käyttäjälle näytetään dialogi, jonka avulla hän pystyy määrittelemään template-elementille asetettavat metaominaisuudet. Kun käyttäjä hyväksyy asettamansa ominaisuudet, luodaan uusi template-elementti, jonka ulkoasuksi asetetaan valitut solut. Tätä samaa dialogia käytetään myös template-elementtien muokkaamiseen.

Laskentataulukkoa edustava template-elementti luodaan, kun näkymätyyppi avataan ensimmäisen kerran muokattavaksi. Sille tallennetaan rivien ja sarakkeiden koot sekä kaikki solut, jotka eivät kuulu millekään muulle template-elementille.

6. TYÖKALUN SUUNNITTELU JA TOTEUTUS

Tässä luvussa kuvataan Trinityyn integroitaville Microsoft Office työkaluille toteutetun Tool-ohjelmistokehyksen arkkitehtuuri sekä tärkeimmät suunnitteluratkaisut. Lopuksi kuvataan siitä erikoistetun Excel-mallinnustyökalun tärkeimmät suunnitteluratkaisut sekä esitellään työn kannalta oleellisten moduulitason toteutusten yksityiskohtia.

6.1 Trinityn mallinnustyökaluille yhteinen Tool-ohjelmistokehys

Trinityn Excel-laajennuksen ensimmäinen versio tehtiin Tampereen teknillisen yliopiston tarjoamalla Ohjelmistotuotannon projektityö -kurssilla. Samalla tehtiin myös ensimmäinen versio Microsoft Word -laajennuksesta. Molemmilla työkaluilla huomattiin olevan samoja vaatimuksia ja toimintoja, jotka päätettiin toteuttaa omana komponenttinaan. Toteutuksen pohjana käytettiin Visio-laajennuksen sen hetkistä toteutusta, josta otettiin mukaan Excel- ja Word-laajennusten vaatimia osia. Toteutetulle komponentille annettiin nimeksi Tool.

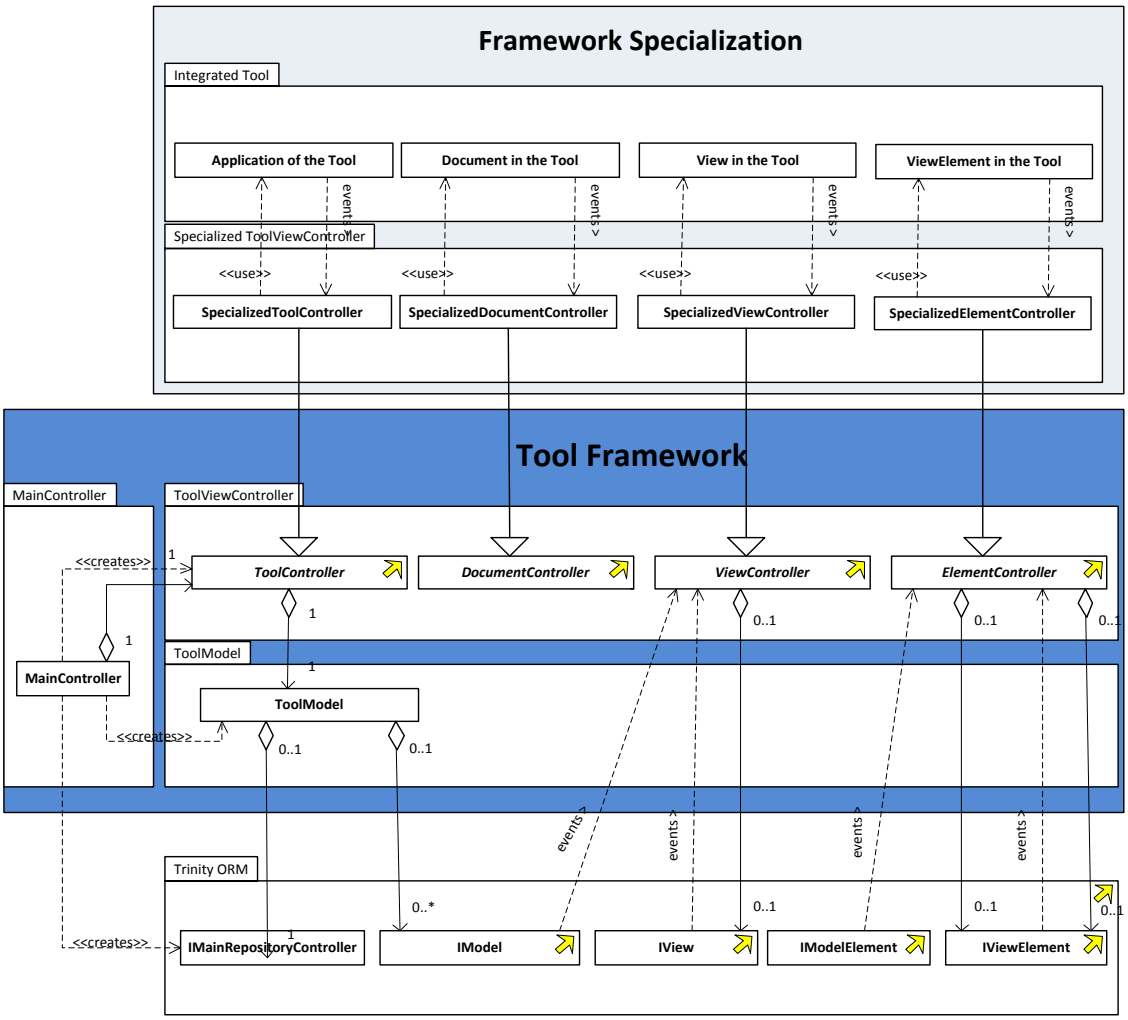
Tool-komponentti on muuttunut huomattavasti sen ensimmäisen toteutuksen jälkeen tarpeiden muuttuessa. Nykyään sitä voitaisiin kutsua muunneltavaksi ohjelmistokehykseksi, koska se tarjoaa yhteisen perustoiminnallisuuden .NET-ympäristössä Trinityyn integroitaville työkaluille. Tätä perustoiminnallisuutta on tarkoitus laajentaa Trinityn työkaluiksi erikoistamalla kehyksen tarjoamia luokkia sekä toteuttamalla muu työkalun vaatima toiminnallisuus.

Ohjelmistokehystä käytetään tällä hetkellä vain Microsoft Office -ohjelmistojen integrointiin, joten se sisältää muutaman Officen rajapinnoista riippuvaisen luokan liittyen Office-ohjelmien valintanauhaan ja kontekstivalikkoon. Mikäli myöhemmin tulee tarvetta integroida muita kuin Office-ohelmistoja, voidaan nämä luokat eriyttää kohtuullisen pienellä vaivalla omaksi komponentikseen ja tehdä Tool-ohjelmistokehyksestä kaikkien .NET-ympäristön ohjelmien integrointiin soveltuva kehys.

6.1.1 Arkkitehtuuri

Tool-ohjelmistokehyksen arkkitehtuurimalliksi valittiin kerrosarkkitehtuuri helpottamaan kehyksen luokkien ryhmittelyä ja tekemään siitä modulaarisempi. Kuva 6.1 esittää Tool-ohjelmistokehyksen arkkitehtuurin korkealla abstraktiotasolla. Kehyksen ryhmittely kerrosarkkitehtuuriksi voidaan ajatella siten, että alimpana kerroksena sijaitsee Trinityn tietokantakomponentti, sen yläpuolella ToolModel-kerros, jonka päällä sijaitsee ToolViewController-kerros. Ylimmäksi kerrokseksi voidaan ajatella integroitava työka-

lu, joka vaatii avukseen työkalukohtaisen erikoistuksen ViewController-kerroksesta. Kehys voidaan myös ajatella Malli-näkymä-ohjain-arkkitehtuurimallina, jossa ToolModel-kerroksen sekä Trinityn tietokantakomponentin luokat edustavat malleja, ViewController-kerros ohjaimia ja integroitavan työkalun komponentit näkymiä.



Kuva 6.1. Tool-ohjelmistokehyksen arkkitehtuuri.

MainController-komponentin tehtävänä on työkalun alustaminen mallinnuskäyttöön. Lisäksi se toimii liityntärajapintana Trinityn agenttiarkkitehtuuriin ja reagoi sieltä saapuvien agenttien pyyntöihin. Näitä ovat muun muassa näkymien avaaminen ja elementtien valitseminen näkymistä. Lisäksi sen tehtävänä on palvelukutsujen lähettäminen agenttien avulla muille ympäristön osille. Näitä ovat esimerkiksi tiedon lähettäminen työkalun ja näkymien sulkemisesta hallintakäyttöliittymälle sekä näkymäelementtien valitseminen muista työkaluista.

ToolModel-kerros sisältää luokkia, jotka pitävät kirjaa työkalun tilaan liittyvistä ominaisuuksista. Näitä ovat muun muassa työkalussa auki olevat näkymät sekä mallit, joiden muutoksia on rekisteröidytty kuuntelemaan. Lisäksi kerros sisältää joukon luokkia, jotka tarjoavat apufunktioita tietokantakomponentin tarjoamien elementtien käsitte-

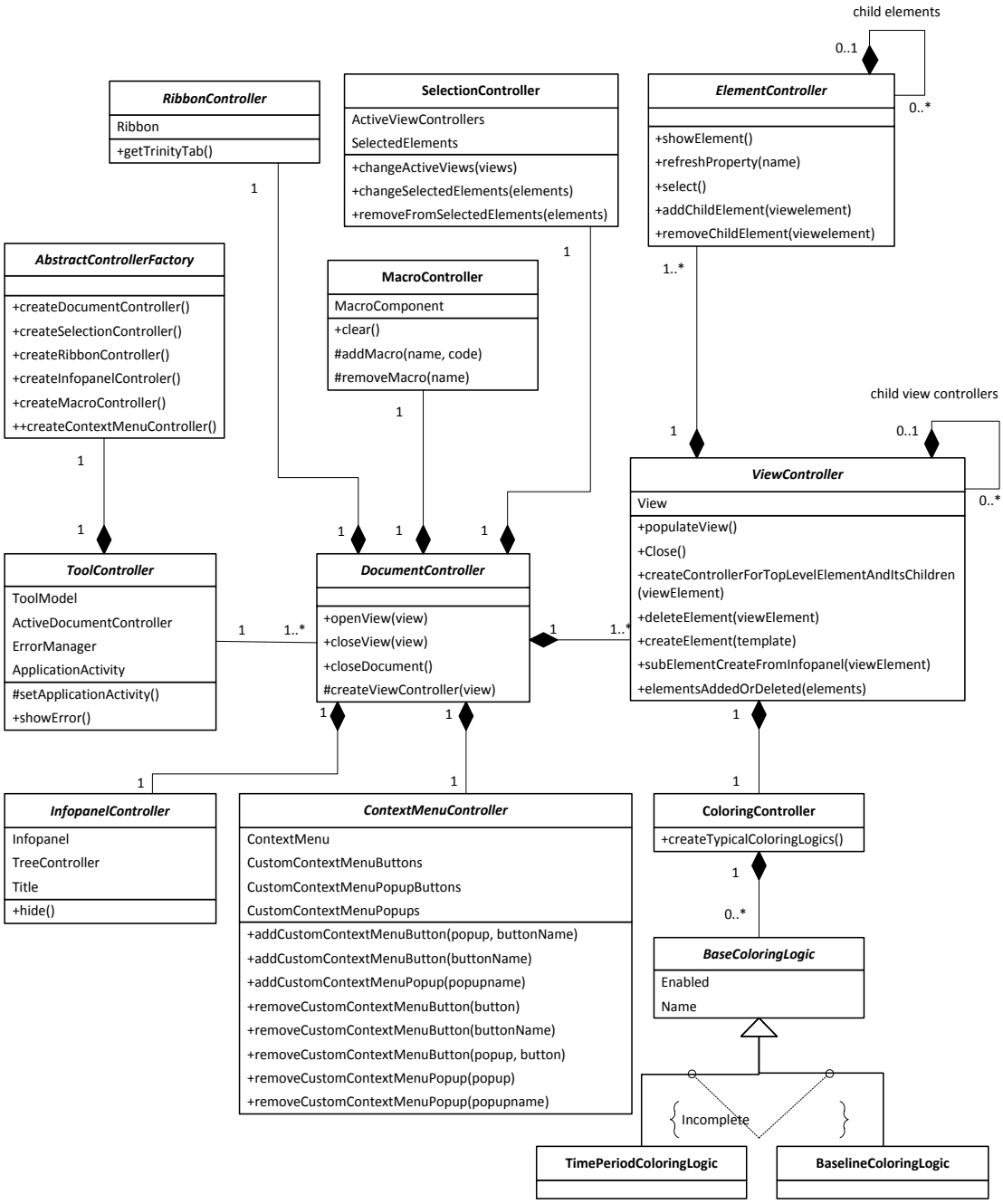
lyyn. Näitä ovat esimerkiksi toiminnot elementtien kytkemiseksi toisiinsa ja tiettyyn listaelementtiin kuuluvien mallielementtien hakeminen. Kaikki malleihin, näkymiin ja muihin Trinityn elementteihin liittyvä tieto on tallessa tietokannassa, jota käsitellään tietokantakomponentin kautta. Tietokantakomponentin käyttö sallitaan suoraan View-Controller-kerroksesta tehokkuussyistä. Kutsuja ei siis välitetä ToolModel-kerroksen kautta, vaikka se pitääkin tallessa tietokantakomponentin ilmentymää (IMainRepositoryController).

ViewControllers-kerros on Tool-kehyksen keskeisin osa. Se sisältää joukon Malli-näkymä-ohjain-arkkitehtuurin ohjaimia, joiden tehtävänä on synkronoida tietokannan ja työkalun käyttöliittymän tilat toistensa kanssa. Se kuuntelee tietokantakomponentin muutoksia ja suorittaa niiden perusteella päivityskäskyjä integroidulle työkalulle. Työkalulta tulevien, käyttäjän toimista aiheutuvien, tapahtumien kuuntelu ja niihin reagoiminen on tarkoitus hoitaa ohjelmistokehyksen erikoistuksessa, koska integroitavat työkalut voivat erota suuresti toisistaan ja tarjota hyvin erilaisia tapahtumia.

6.1.2 Kehyksen tarjoamat palvelut

Kuva 6.2 esittää luokkakavion Tool-ohjelmistokehyksen ViewController-kerroksen tärkeimmistä luokista. ToolController-luokan tehtävänä on hallita integroitavaa sovellusta. Perustoteutus tarjoaa mekanismin virheiden näyttämiseksi, pääsyn ToolModel-luokkaan ja sitä kautta Trinityn tietokantakomponenttiin sekä pääsyn aktiiviseen DocumentController-olioon. ToolController-olio omistaa AbstractControllerFactory-olion, joka toteuttaa nimensä mukaisesti Abstrakti tehdas -suunnittelumallin. Sen avulla luodaan muita tarvittavia ohjaimia, joille voi tai täytyy antaa toteutus tapauksesta riippuen kehyksen erikoistuksessa.

DocumentController-luokan tehtävänä on hallita yksittäistä työkalun dokumenttia. Siihen liittyy dokumentin käyttöliittymän osia hallitsevat RibbonController-, MacroController-, ContextMenuController- ja InfopanelController-luokka. RibbonController-luokan tehtävänä on hallita dokumentin valintanauhan Trinity-välilehteä. MacroController-luokan avulla voidaan lisätä ajonaikaisesti uusia makroja integroitavaan sovellukseen sekä poistaa niitä. ContextMenuController-luokan avulla voidaan lisätä sovelluksen kontekstivalikkoon uusia nappeja sekä ponnahdusvalikoita. Nämä kolme luokkaa ovat sidoksissa Microsoft Officeen ja toimivat vain siihen kuuluvien sovellusten kanssa. InfopanelController-luokka tarjoaa pääsyn Trinityn informaatiopaneeliin ja se reagoi siellä tehtäviin toimintoihin. Lisäksi jokainen DocumentController-olio omistaa SelectionController-olion, joka pitää kirjaa dokumentissa kulloinkin valittuina olevista näkymistä ja näkymäelementeistä.



Kuva 6.2. Tool-ohjelmistokehyksen ViewControllers-kerroksen luokkakaavio.

ViewController-luokan tehtävänä on hallita yksittäistä näkymää työkalussa. Se tarjoaa perustoteutukset näkymän avaamiselle ja elementtien poistamiselle. ViewControlleriin voi hallita koosteista näkymää, johon liittyy lapsinäkymiä. Tällöin se tietää lapsinäkymien ViewController-oliot. ViewController omistaa ColoringController-olion, jonka avulla hallitaan näkymän muutosvärityksiä. Lisäksi se omistaa ElementController-oliot, jotka luodaan jokaiselle näkymään kuuluvalle näkymäelementille. ElementControllerin perustoteutus tarjoaa toteutukset lapsielementtien lisäämiseksi ja poistami-

seksi. Listaelementtien tapauksessa ne hoitavat myös tarvittavan suhteen luomisen ja poistamisen mallielementtien välillä.

6.1.3 Kehyksen erikoistamisrajapinta

Tool-ohjelmistokehys sisältää paljon luokkia, jotka sisältävät abstrakteja metodeita. Niille on pakko antaa toteutus kehyksen erikoistuksissa. Suurin osa perustoteutuksen tarjoamista metodeista on määritelty virtuaalisiksi, jolloin niille voidaan antaa uusi toteutus erikoistuksessa, mikäli perustoteutusta halutaan laajentaa tai se ei ole sopiva. Seuraavaksi kuvataan tärkeimmät kohdat kehyksen erikoistamisrajapinnasta.

MainController-luokka sisältää tehdasmetodit ToolModel- ja ToolController-olion luomiseksi. Erikoistuksen täytyy antaa näille työkalukohtaiset toteutukset. Lisäksi ToolModel-luokan erikoistukseen pitää antaa toteutus työkalukohtaisen näkymäelementin tietojen rekisteröimiseksi tietokantakomponentille, jotta se osaa ladata ja tallentaa työkalukohtaista tietoa.

ToolController-luokka pitää erikoistaa ja liittää integroitavan työkalun ajossa olevaan prosessiin, koska erikoistuksen täytyy reagoida työkalun sulkemiseen ja kutsua tarvittavia siivousmetodeita. AbstractControllerFactory-luokalle täytyy toteuttaa työkalukohtainen konkreettinen tehdas, jotta työkalu saadaan alustettua käyttökuntoon.

DocumentController-luokan erikoistukselle täytyy toteuttaa metodit näkymän avaamiselle, sulkemiselle ja aktivoimiselle sekä dokumentin sulkeminen. ViewController-luokan erikoistuksen täytyy toteuttaa näkymän avaaminen, jolloin luodaan tarvittavat ElementController-oliot ja asetetaan näkymä käyttökuntoon. Erikoistettujen ElementController-olioiden luomiseksi täytyy toteuttaa tehdasmetodi, jonka avulla niiden luominen onnistuu. Lisäksi tarvitaan toteutukset elementtien luomiseksi annetun template-elementin perusteella, tarvittavien työkalukohtaisten toimenpiteiden suorittaminen infopaneelista luoduille lapsielementeille ja reagoiminen tietokantakomponentin ilmoitukseen muualla luoduista tai poistetuista elementeistä.

ElementController-luokan erikoistuksessa pitää toteuttaa elementin näyttäminen näkymässä, jota ViewController-olio kutsuu näkymän avaamisen jälkeen ja uuden elementin luomisen yhteydessä. Lisäksi pitää toteuttaa päivitysmetodi, jota kutsutaan mallielementin ominaisuuden arvon muuttumisen jälkeen sekä metodi elementin valitsemiseksi työkalussa.

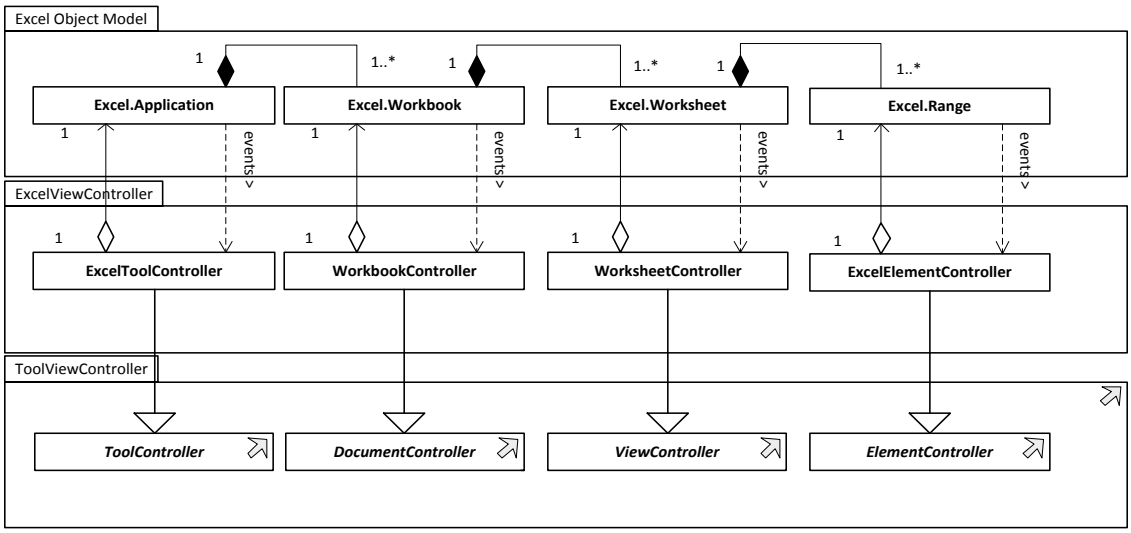
6.2 Excel-laajennuksen toteuttaminen

Työkalu toteutettiin Exceliin sovellustason laajennuksena [33] käyttäen Excelin oliomallia. Laajennus tehtiin erikoistamalla Tool-ohjelmistokehys. Kehyksen tarjoamille luokkien perustoteutuksille tehtiin Excel-spesifiset erikoistukset, jotka käyttävät oliomallin tarjoamia luokkia.

6.2.1 Tool-ohjelmistokehyksen erikoistaminen

Laajennuksen arkkitehtuuri noudattaa Tool-ohjelmistokehyksen arkkitehtuuria ja sen komponentit nimettiin saman nimeämiskäytännön mukaan. MainController-komponentti erikoistettiin ExcelMainController-komponentiksi, joka sisältää liityntärajapinnan ajossa olevaan Excel-prosessiin. Lisäksi Model- ja ViewController-kerroksista tehtiin vastaavat ExcelModel- ja ExcelViewController-moduulit. Erikoistetut komponentit toteuttavat kaikki Tool-kehyksen erikoistamisrajapinnan vaatimat metodit.

ExcelViewController-moduulin luokat käyttävät oliomallia näkymän roolissa. Ne suorittavat tarvittavat käyttöliittymän päivitykset niiden avulla sekä kuuntelevat käyttäjän tekemistä toiminnoista aiheutuvia tapahtumia. Kuva 6.3 esittää luokkakaaviona ExcelViewController-moduulin tärkeimpien luokkien suhtautumisen Tool-kehyksen kantaluokkiin sekä oliomallin luokkiin.



Kuva 6.3. Luokkakaavio ExcelViewController-moduulin keskeisimpien luokkien suhteista Tool-kehyksen vastaaviin kantaluokkiin sekä oliomallin tarjoamiin luokkin.

AbstractControllerFactory-luokka erikoistettiin ExcelControllerFactory-luokaksi, johon toteutettiin luontimetodit tarvittaville Excel-laajennuksen ohjaimille. ToolController-luokka laajennettiin ExcelToolController-luokaksi, jonka tehtävänä on kuunnella oliomallin Application-luokan tarjoamia tapahtumia. Se reagoi muun muassa Excel-sovelluksen sulkeutumiseen ja uusien työkirjojen avaamiseen.

WorkbookController-luokka toteuttaa DocumentController-luokan abstraktit metodit ja se on sidottu oliomallin Workbook-olioon. Sen tehtävänä on reagoida laskentataulukoiden sulkemisiin sekä luoda Excel-laajennukseen avattaville näkymille uusia laskentataulukoita. Myös SelectionController-luokka laajennettiin, koska työkalussa pitää tallettaa tieto edellisestä valinnasta, jotta voidaan tarkastella onko valitun alueen muotoiluja muutettu.

WorksheetController-luokka laajentaa ViewController-luokan tarjoamalla Excelin laskentataulukkoon liittyviä toimintoja. Se liitetään oliomallin Worksheet-olioon,

jolta se kuuntelee muun muassa laskentataulukossa tapahtuvia muutoksia. Jokainen WorksheetController-olio sisältää InternalChangeHandler ja ExternalChangeHandler-oliot, joista ensimmäisen tehtävänä on tarkistaa, mitä käyttäjä teki ennen Change-tapahtuman saapumista ja suorittaa tarvittavat toimenpiteet. Jälkimmäinen puolestaan reagoi tietokantakomponentin ilmoituksiin näkymäelementtien lisäyksistä, poistoista ja muutoksista, jonka jälkeen se päivittää laskentataulukon tilan vastaamaan muuttunutta tilannetta.

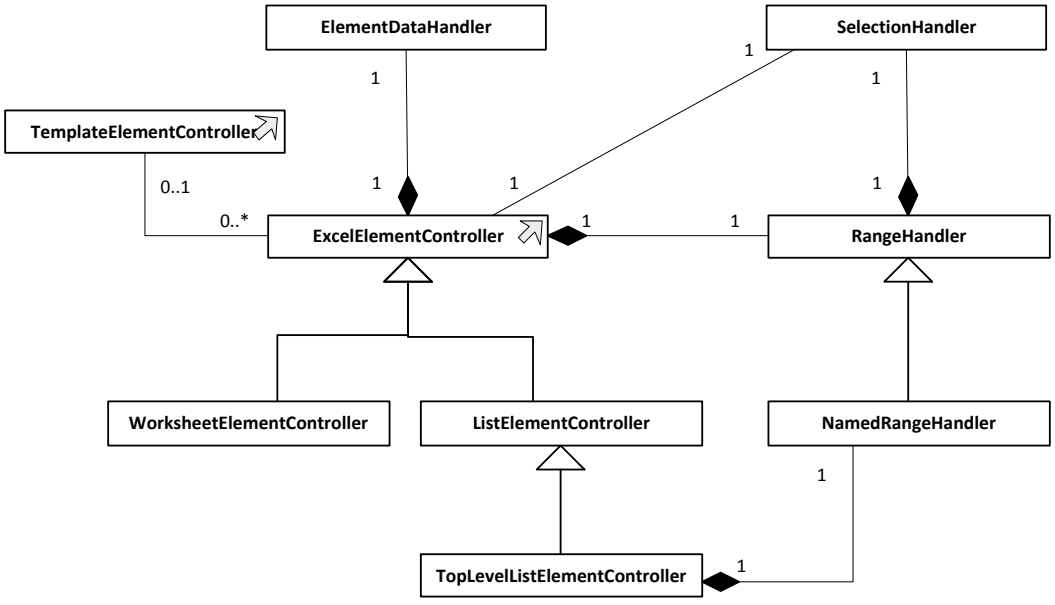
ExcelElementController-luokka erikoistaa ElementController-luokan ja se liitetään oliomallin Range-luokkaan. ExcelElementController-luokka, sen aliluokat ja näihin läheisesti liittyvät muut luokat on kuvattu tarkemmin seuraavassa aliluvussa 6.2.2.

6.2.2 ElementControllers-moduuli

Tool-ohjelmistokehysen tarjoaman ElementController-luokan Excel-erikoistus tarvitsi huomattavasti enemmän toteutusta kuin muut kehyksestä laajennetut luokat. ExcelElementController-luokasta, sen aliluokista ja näihin läheisesti liittyvistä luokista muodostettiin oma koodimoduulinsa. Kuva 6.4 esittää tämän ElementControllers-moduulin luokat sekä niiden väliset suhteet.

ExcelElementController-luokasta on erikoistettu WorksheetElementController-, ListElementController- ja TopLevelListElementController-luokka. Näistä ensimmäinen hallitsee laskentataulukkoa edustavaa näkymäelementtiä, toinen näkymäelementillä lapsena olevaa listaelementtiä ja viimeinen ylimmän tason listaelementtiä eli esimerkiksi raportissa olevaa listaa elementeistä, joilla ei ole yhteistä isä-elementtiä.

ElementDataHandler-luokan tehtävänä on hallita näkymäelementtiin liittyvää Excel-kohtaista tietoa. Sen tehtävänä on huolehtia elementtiin liittyvien solujen tietojen (CellInfo) ja suodatinten alustamisesta, näyttämisestä sekä muista niiden hallintaan liittyvistä tarpeellisista toiminnoista.

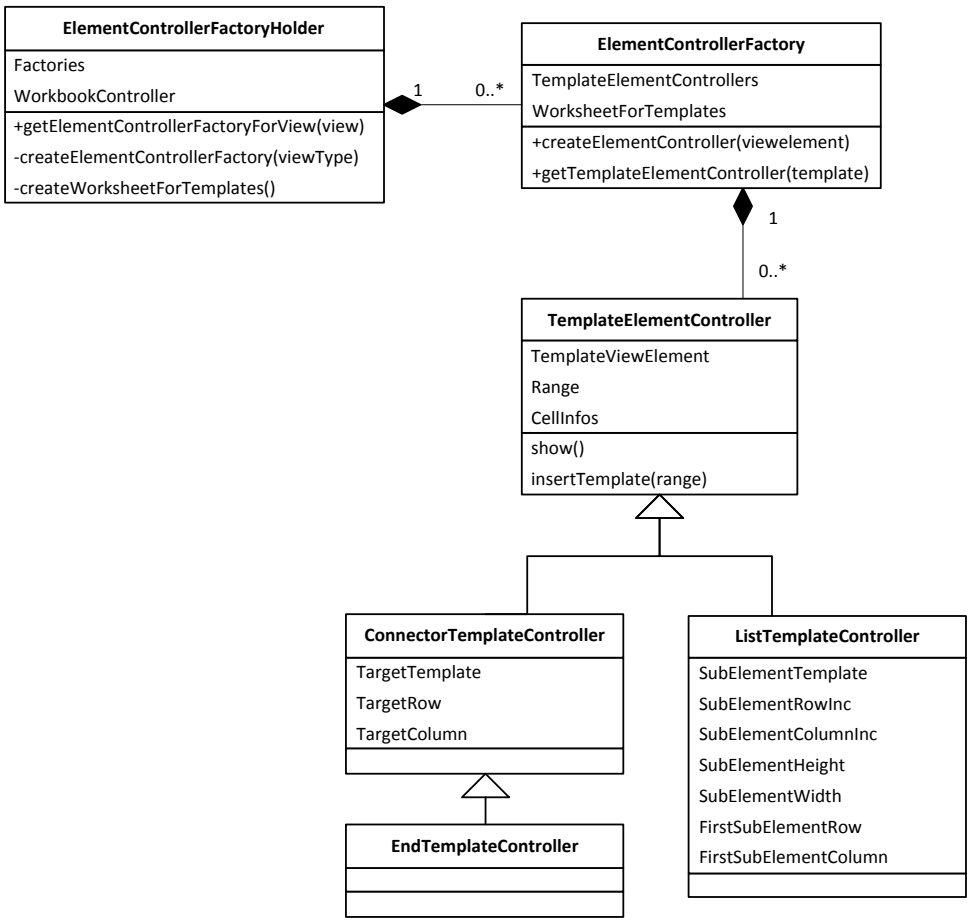


Kuva 6.4. ElementControllers-moduulin luokkakaavio.

RangeHandler-luokka hoitaa varsinainen ExcelElementController-olion sitominen oliomallin Range-olioon. Se sisältää kaikki toiminnot näkymäelementille kuuluvan solualueen käsittelyyn. Korkeimman tason elementit sidotaan NamedRange-olioon, koska siltä saadaan tapahtumia elementin solualueen valinnan vaihtumisesta ja valinnan poistumisesta. Kaikki elementit eivät käytä NamedRange-olioita, koska lapsielementit sijaitsevat samalla solualueella isä-elementtiensä kanssa, joten niiden valinnan vaihtumisesta tulisi monta päällekkäistä tapahtumaa. NamedRangeHandler-luokat hoitavat NamedRange-olioiden hallitsemisen ja niiden valintoihin liittyvien tapahtumien kuuntelut. SelectionHandler-olion tehtävänä on tarkistaa kuuluko valittu alue elementin solualueeseen.

6.2.3 TemplateElementControllers-moduuli

Kuva 6.5 esittää TemplateElementController-luokan ja siihen läheisesti liittyvät muut luokat. ExcelToolController-olio omistaa ElementControllerFactoryHolder-olion, joka pitää tallessa, sekä luo tarvittaessa uusia, näkymätyyppeihin liittyviä ElementControllerFactory-olioita. Jokaiseen WorksheetController-olioon liittyy sen näkymän näkymätyyppiä vastaava ElementControllerFactory-olio, jonka avulla se luo näkymäelementeille ElementController-oliot.



Kuva 6.5. TemplateElementController ja siihen liittyvät luokat.

Jokaista template-elementtiä vastaa TemplateElementController-olio, jonka tehtävänä on toimia kääreenä Trinityn tietokannassa olevalle template-elementille. Niiden avulla saa selvitettyä template-elementtiin liittyviä Excel-kohtaisia tietoja. Jokaiselle template-elementille annetaan viittaus solualueeseen, johon asetetaan template-elementin ulkoasu. TemplateElementController-olioita voi pyytää asettamaan ulkoasunsa haluttuun solualuseen. Tämä nopeuttaa elementtien luomista ja lataamista tietokannasta, koska kaikki yhteen template-elementtiin liittyvät tiedot saadaan asetettua kerralla eikä jokaisen solun tietoja joudu asettamaan yksitellen.

TemplateElementControllerista on peritty lista-, suhde- ja suhteenpää-template-elementeille omat luokkansa. Niiden tehtävänä on helpottaa kyseisen tyyppisten elementtien käsittelyä. Esimerkiksi lista-elementit saavat kysyttyä lapsien lisäämiseen ja poistamiseen tarvittavia tietoja.

6.2.4 Reporting-moduuli

Raportointitoiminnot toteutettiin omaan Reporting-koodimoduuliinsa, joka sisältää Malli-näkymä-ohjain-arkkitehtuurin mukaiset Model-, View- ja ViewController-alimoduulit. Kyseinen moduuli tullaan tulevaisuudessa todennäköisesti siirtämään osaksi Tool-ohjelmistokehystä, jotta sen tarjoamia toimintoja voitaisiin käyttää myös muissa Trinityn työkaluissa.

Model-alimoduuli sisältää ReportManager- ja FilterHelpers-luokat. ModelManagerin tehtäviin kuuluu näkymän asettaminen raportiksi ja elementtien suodatus asetettujen suodattimien perusteella. FilterHelpers-luokka sisältää staattisia apufunktioita suodatinten käsittelyn helpottamiseksi.

View-alimoduuli sisältää Windows Form -luokan toteuttavat AddFilterDialog-, CreateReportDialog- sekä ManageFiltersDialog-luokat. Kyseiset luokat toteuttavat käyttöliittymät suodattimien lisäämiselle ja muokkaamiselle, raporttien luomiselle malleista ja näkymistä sekä listaelementtien suodatinten näyttämisen, priorisoinnin ja poistamisen.

ViewController-alimoduuli sisältää ohjaimet View-alimoduulin dialogeille. Jokaista dialogia vastaa oma ohjaimensa, jonka tehtävänä on hakea tietoja järjestelmästä dialogille sekä asettaa käyttäjän dialogiin syöttämät tiedot raporteille ja listaelementeille. Alimoduulin luokat on nimetty dialogien mukaan vastaavalla tavalla AddFilterDialogController-, CreateReportDialogController- sekä ManageFiltersDialogController-luokaksi.

6.2.5 FormattingAdapters-moduuli

Oliomallin Range-luokan tarjoamien muotoilujen ja niistä tietokantaan talletettavien tietojen (FormattingInfo ja FontInfo) välillä käytetään Sovitin-suunnittelumallia helpottamaan muotoilujen vertailua, tietokantaan talletusta sekä solualuille asettamista.

Laajennuksessa on tallessa työkirjan oletussolutyyliä vastaava sovitin-olio, johon muuttuneiden solujen muotoiluja verrataan. Soluista ei tarvitse erikseen tallettaa muotoiluja, mikäli ne vastaavat oletussolutyyliä.

6.3 Toteutuksen haasteita

Excel-laajennuksen toteutuksessa haasteita aiheuttivat Excelin oliomalli sekä Trinityn hajautettu mallinnusympäristö. Oliomalli tarjoaa heikosti tukea laskentataulukon muutosten havaitsemiseksi. Se tarjoaa käytännössä vain Change- ja SelectionChange-tapahtumat, joiden avulla voidaan selvittää muutokset ja reagoida niihin. Lisäksi toimintojen kumoaminen ja uudelleen tekeminen Excelissä aiheutti haasteita. Myös Trinityn hajautettu ympäristö vaati mallien yhtäaikaisen käsittelyn huomioon ottamista.

6.3.1 Muutosten havaitseminen Excelin oliomallin avulla

Change-tapahtuma laukaistaan, kun laskentataulukossa muutetaan jonkin solualueen arvoja, lisätään soluja tai poistetaan niitä. Tapahtuman mukana tulee viittaus muuttuneeseen solualueeseen, mutta ei minkälaista tietoa muutoksen tyypistä. Solualueen arvojen muuttuminen on melko yksinkertaista tarkastaa, mutta esimerkiksi laskentataulukosta poistettujen rivejä tai sarakkeiden tapauksessa saadaan vain Change-tapahtuma ja viittaus lisättyyn tai poistettuun solualueeseen, eikä mitään muuta tietoa.

Solualueen muutokset täytyy selvittää vertaamalla laskentataulukon aiempaa tilaa muutoksen jälkeiseen tilaan. Valinnan vaihtuessa laskentataulukossa valitut elementit sekä valittu alue tallennetaan, koska muutoksia ei tarvitse tutkia kuin näiltä elementeiltä. Arvojen muutoksista tutkitaan tyhjennettiinkö elementtien koko solualue, jolloin valitut elementit poistettiin. Jos arvoja muutettiin, tallennetaan uudet arvot elementeille.

Solualueiden kopiointi, leikkaaminen ja liimaaminen sekä hiirellä siirtäminen ovat hieman monimutkaisempia tapauksia. Kopioinnin havaitsemiseksi pitää tutkia oliomallin Application-olion CutCopyMode-ominaisuuden muutoksia, kun valintaa vaihdetaan laskentataulukossa. Jos CutCopyMode-ominaisuuden arvoksi vaihtuu xlCopy tai xlCut, tallennetaan valitut elementit ja tieto niiden kopioimisesta tai leikkaamisesta. Seuraavan muutostapahtuman saapuessa on tutkittava vastaako muuttuneen alueen sisältö kopioitua tai leikattua aluetta. Solualueen siirtämisestä saadaan kaksi Change-tapahtumaa, joista ensimmäinen viittaa solualueeseen ennen siirtämistä ja jälkimmäinen solualueeseen siirtämisen jälkeen. Elementtien hiirellä siirtämisen havaitsemiseksi on tutkittava valittujen elementtien sijaintia tietokannassa ja verrattava niitä elementin nykyisen solualueen sijaintiin.

Oliomalli ei tarjoa laskentataulukon muotoilujen muutoksista ollenkaan tapahtumia. Ainoa keino selvittää muutosten muuttuminen on laskentataulukossa viimeksi valittuna olleen alueen muotoilujen vertaaminen ennen uutta valintaa ja sen jälkeen. Tällä tavoin ei kuitenkaan saada selville kaikkia muutoksia. Esimerkiksi rivien ja sarak-

keiden kokoja sekä laskentataulukon asetuksia voidaan muuttaa ilman, että laskentataulukon valintaa täytyy muuttaa.

6.3.2 Toimintojen peruminen ja uudelleen tekeminen

Excel tyhjentää koko undo-puskurinsa, kun laskentataulukon tilaa muutetaan oliomallin kautta. Oliomallin kautta ei ole pääsyä undo-puskuriin ja ainoa tapa määritellä toimintojen perumisessa sekä uudelleen tekemisessä tapahtuvat toiminnot on määritellä makrot, joita kutsutaan kyseisissä toiminnoissa. Excel-laajennukselle lisättiin doUndo ja doRedo makrot, jotka kutsuvat Excel-laajennukseen toteutetun UndoRedoController-olion metodeita.

UndoRedoController sisältää kutsujonon, johon tallennetaan käyttäjältä saapuvat toimintojen perumiset ja uudelleen tekemiset. Se suorittaa jonossa olevat toiminnot järjestyksessä kutsumalla tietokantakomponentin vastaavia palveluita. Tietokantakomponentti palauttaa tietokannan tilan viimeisintä transaktiota vastaavaan tilaan ja tiedottaa muutoksista tapahtumilla. Excel-laajennus reagoi näihin tapahtumiin saman lailla kuin ulkoisiin muutoksiin eli asettamalla laskentataulukon vastaamaan tietokannan tilaa.

6.3.3 Hajautettu mallinnusympäristö

Excel sallii laskentataulukon solujen muuttamisen oliomallin kautta samaan aikaan kun käyttäjä muokkaa toisen solun arvoa. Mikäli käyttäjä muokkaa samaa solua, jonka arvoa ollaan muuttamassa oliomallin kautta, oliomallin kautta tehdyt muutokset asetetaan vasta kun käyttäjä on saanut muutoksensa valmiiksi. Jos käyttäjä avaa Excelistä jonkin dialogin, lukitaan sovelluksen pääsäie (mainthread) siksi aikaa, kun dialogi suljetaan. Tämän jälkeen suoritusta jatketaan normaalisti.

Edellä kuvattujen ominaisuuksien vuoksi hajautetusta ympäristöstä ei aiheutunut suuria ongelmia työkalun toteutuksessa. Suurimmat ongelmat aiheutuvat siitä, kun käyttäjät lisäävät uusia elementtejä samaan kohtaan laskentataulukkoa. Excel-näkymäelementtien solualue määriteltiin siten, että vain elementin lapsielementit saavat olla samalla alueella sen kanssa. Samanaikainen lisääminen luo kuitenkin elementit päällekkäisille solualuille. Tämä ongelma voidaan kuitenkin kiertää tutkimalla ElementController-oliota luotaessa uuden näkymäelementin solualueen ja muiden ylimmän tason näkymäelementtien solualueiden leikkauksia. Mikäli elementtien solualueet leikkaavat, siirretään viimeiseksi luotu näkyelementti seuraavaan vapaaseen kohtaan laskentataulukossa.

7. ARVIOINTI

Tässä luvussa arvioidaan diplomityössä toteutetun Excel-mallinnustyökalun toteutusta. Työssä esitettyjen ratkaisujen vastaavuutta verrataan luvussa 4 esiteltyihin vaatimuksiin ja pohditaan valittujen ratkaisujen hyviä sekä huonoja puolia. Lopuksi pohditaan työkalun tulevaisuutta.

7.1 Toteutuksen arviointi

Seuraavissa taulukoissa arvioidaan työssä esitettyjen ratkaisujen vaatimuskattavuutta asteikolla {-, 0, +}. Asteikon miinus-merkki tarkoittaa, ettei vaatimus täyttynyt ollenkaan, nolla tarkoittaa osittaista täyttymistä ja plus-merkki tarkoittaa vaatimuksen täyttymistä. Arviot ovat osin subjektiivisia, mutta ne pyritään perustelemaan. Ne vaatimukset, jotka eivät täysin toteutuneet, käsitellään tarkemmin.

Työn tavoitteena oli integroida Trinity-ympäristöön taulukkomuotoinen raportointi- ja mallinnustyökalu. Tavoitteeseen päästiin toteuttamalla Microsoft Exceliin laajennus, joka toteuttaa pääosin kyseiselle työkalulle asetetut vaatimukset. Trinityyn integroitaville Microsoft Office sovelluksille toteutettiin ohjelmistokehys helpottamaan niiden integrointia sekä tarjoamaan Microsoft Word- ja Excel-laajennukselle yhteinen toiminnallisuus.

Taulukko 7.1 esittää raportoinnin vaatimusten toteutumisen. Asetetut raportointivaatimukset saatiin toteutettua hyvin. Työkalun avulla voidaan generoida raportteja ympäristön malleista ja näkymistä. Raporttiin tulevia elementtejä pystyy suodattamaan erilaisilla suodattimilla melko monipuolisesti. Lisäksi muutokset päivittyvät raportteihin automaattisesti Trinityn tarjoamien mekanismien avulla.

Taulukko 7.1. Raportoinnin vaatimusten toteutuminen.

Vaatimuksen tunniste ja nimi	**Toteutuminen**
V01 Raporttinäkymän generointi malleista ja toisista näkymistä	+
V02 Elementtien suodattaminen	+
V03 Muutosten automaattinen päivittyminen raporttiin	+

Taulukko 7.2 esittää mallinnustoimintojen vaatimusten toteutumisen. Mallinnustoiminnoista suurin osa saatiin toteutettua vaatimusten mukaisesti. Työkalun suorituskyky ei ole erityisen hyvä ja esimerkiksi elementtien luomisen ja kopioinnin yhteydessä aiheutuu viivettä, kun elementit tallennetaan tietokantaan. Osa suorituskykyongelmista johtuu Trinityn tietokantapohjaisuudesta, mutta myös Excel-laajennuksen suorituskyky-

vyssä olisi parantamisen varaa. Kaikkia laskentataulukon muotoiluun liittyviä tietoja ei kirjoitushetkellä tallenneta tietokantaan. Esimerkiksi kuvaajat, kuvat eivätkä muutkaan objektit tallennu ollenkaan näkymän tietoihin. Lisäksi elementtien muutosten korostaminen on kirjoitushetkellä toteutettu vaillinaisesti.

Taulukko 7.2. Mallinnustoimintojen vaatimusten toteutuminen.

Vaatimuksen tunniste ja nimi	**Toteutuminen**
V04 Ei ylimääräisiä vaiheita työhön	+
V05 Riittävä suorituskyky	0
V06 Toimintojen kumoaminen ja uudelleen tekeminen	+
V07 Mallielementin ominaisuuksien muokkaaminen	+
V08 Näkymän muotoilujen muutosten tallentuminen	0
V09 Uusien elementtien luominen	+
V10 Viittaavien näkymäelementtien luominen	+
V11 Elementtien poistaminen	+
V12 Elementtien kopiointi työkalu	+
V13 Elementin kopiointi toisesta eri työkalusta	+
V14 Lapsielementtien lisääminen	+
V15 Lapsielementtien poistaminen	+
V16 Lapsielementtien siirtäminen	+
V17 Elementtien luominen valitusta alueesta	+
V18 Tiedon sitominen mallielement	+
V19 Navigointi ensisijaiseen näkymäelementtiin	+
V20 Elementtien muutosten korostaminen	0

Taulukko 7.3 esittää hajautetun mallinnusympäristön vaatimusten toteutumisen. Nämä vaatimukset saatiin toteutettua melko hyvin. Muutosten päivittäminen oli melko yksinkertaista toteuttaa, koska Trinityn tietokantakomponentti tarjoaa tälle hyvin tukea. Muiden käyttäjien tekemien muutosten korostamisen toteuttamisessa ei ole erityisiä teknisiä haasteita ja se voidaan toteuttaa samaan tapaan kuin V20 Elementtien muutosten korostaminen. Sitä ei kuitenkaan ole toteutettu työn kirjoitushetkellä, koska erilaiset muutosten korostamiset ovat muuttumassa Trinityssä.

Taulukko 7.3. Hajautetun mallinnusympäristön vaatimusten toteutuminen.

Vaatimuksen tunniste ja nimi	**Toteutuminen**
V21 Mallielementtien muutosten päivittäminen	+
V22 Näkymän muutosten päivittäminen	+
V23 Muiden käyttäjien tekemien muutosten korostaminen	-

Taulukko 7.4 esittää työkalun laajentamisvaatimusten toteutumisen. Uusien näkymätyyppien lisääminen onnistuu ilman koodiin tehtäviä muutoksia. Sille ei ole kuitenkaan toteutettu käyttöliittymää ja se vaatii tietojen lisäämistä suoraan Trinityn tietokantaan. Tämä ei kuitenkaan ole suuri ongelma, koska näkymätyyppien lisäämistä ei ole tarkoitettu peruskäyttäjien tehtäväksi. Template-elementtejä pystyy lisäämään, poistamaan ja muokkaamaan, mutta tämä toiminto on piilotettu peruskäyttäjiltä.

Taulukko 7.4. Laajentamisvaatimusten toteutuminen.

Vaatimuksen tunniste ja nimi	**Toteutuminen**
V24 Uusien näkymätyyppien ja raporttipohjien lisääminen ilman koodiin tehtäviä muutoksia	0
V25 Template-elementtien lisäys ja muokkaus	+

7.2 Tulevaisuus ja parannusehdotuksia

Trinityn tietokannan suorituskykyongelmista, erityisesti elementtien luomisen yhteydessä, on keskusteltu ja luomisia yritetään lähitulevaisuudessa optimoida muuttamalla luomisoperaatiot asynkronisiksi niissä tapauksissa, missä voidaan. Myös Excel-laajennuksen yleistä suorituskykyä tullaan parantamaan selvittämällä hitaat kohdat koodin profilointityökalulla ja optimoimalla löytyneitä kohtia. Lisäksi Excel-laajennukseen toteutetaan lähitulevaisuudessa puuttuvat muutosten korostamiset sekä hiotaan toimintoja paremmiksi ja parannetaan työkalun käytettävyyttä. Tämän jälkeen työkalu annetaan teollisuusyhteistyökumppanillemme testikäyttöön.

Työkaluun tullaan tulevaisuudessa lisäämään uusia näkymätyyppejä tarpeen mukaan nykyisille sekä tuleville mallinnuskielille. Lisäksi laskentataulukon kuvaajille ja kuville tullaan toteuttamaan sovittimet, joiden avulla myös ne saadaan tallennettua Trinityn tietokantaan.

Laskentataulukon muotoilujen muuttumisen havaitsemisessa olevan ongelman voisi kiertää tallentamalla työkirjan kokonaisuudessaan tietokantaan. Näkymäelementit sisältäisivät vain tiedon siitä, missä kohtaa ne sijaitsevat, ja kaikki muotoilut tulisivat tallennetulta työkirjalta. Tällöin työkirjasta saisi tallennettua muotoilujen lisäksi kaikki kuvat, makrot sekä objektit. Myös näkymien avaaminen nopeutuisi, koska käyttäjät voisivat ladata työkirjan ja avata sen luettavaksi eikä elementtien tietoja tarvitsisi ladata yksitellen näkymiin ennen kuin näkymän pystyy lukemaan. Näkymäelementtien tiedot voisi ladata vasta siinä vaiheessa, kun käyttäjä haluaa muokata näkymää. Ongelmana tässä olisi näkymän muotoilujen muuttaminen samanaikaisesti muiden käyttäjien kanssa. Excel mahdollistaa suojattujen työkirjojen yhdistämisen, mutta suojatussa tilassa monet toiminnot on estetty. Esimerkiksi työkirjan kuvia, kuvaajia eikä objekteja voi lisätä tai poistaa. Jos kokonainen työkirjoja tallennettaisiin aina muotoiluja muutettaessa, muut näkymää samaan aikaan käsittelevät käyttäjät joutuisivat lataamaan muuttuneen työkirjan jokaisen pienen muutoksen jälkeen. Tämän saisi kierrettyä tallentamalla

muutokset tietokantaan vasta, kun käyttäjä on saanut tehtyä kaikki haluamansa muutokset ja hyväksyy ne.

Näkymätyyppien ja template-elementtien tekemistä pitäisi yksinkertaistaa sekä tarjota myös loppukäyttäjille mahdollisuus luoda haluamansalaisia näkymätyyppejä. Tällöin työkalun käyttökohteet kasvaisivat paljon, koska käyttäjät voisivat käsitellä tietoja juuri siinä muodossa kuin haluavat. Myös tietojen lukemista malleihin voisi parantaa tarjoamalla mahdollisuus määritellä laskentataulukossa olevalle tiedolle rakenne, jonka jälkeen koko laskentataulukon sisällön voisi lukea tiedot kerralla mallielementeiksi ja niiden ominaisuuksiksi.

8. YHTEENVETO

Tässä työssä suunniteltiin ja toteutettiin raportointi- ja mallinnustyökalu Microsoft Excel laajennuksena hajautettuun Trinity-ympäristöön. Laajennuksen avulla ympäristön mallien tietoja voidaan esittää ja käsitellä taulukkomuodossa. Se mahdollistaa erilaisten raporttien generoinnin Trinityn malleista, mihin päivitetään muutokset automaattisesti. Lisäksi raporttiin generoitavia elementtejä voidaan suodattaa erilaisilla kriteereillä.

Toteutetun työkalun avulla on mahdollista muokata mallien tietoja Excel-näkymien kautta. Malleihin on mahdollista lisätä uusia mallielementtejä sekä niitä voidaan poistaa ja niiden ominaisuuksia voidaan muuttaa. Työkalu soveltuu hyvin tietojen syöttämiseen, koska erityisesti pitkien kuvausten syöttäminen on nopeampaa tekstimuodossa kuin graafisilla työkaluilla.

Toteutettu Excel-työkalu on hyvä lisä Trinity-ympäristöön, missä sitä voidaan käyttää yhdessä muiden ympäristön työkalujen kanssa. Tiedot päivittyvät työkalujen välillä Trinityn tarjoamien mekanismien avulla. Esimerkiksi Trinityyn toteutetulla Microsoft Visio -laajennuksella voidaan piirtää luokkakaavio, joissa esitetään mallinnettavan järjestelmän yleiskuva. Tämän jälkeen kaaviossa olevista elementeistä voidaan generoida näkymäelementtejä Excel-näkymään, ja tarkentaa niiden tietoja siellä. Toinen vaihtoehto on kuvata ensin käsitteitä ja niiden tärkeimpiä ominaisuuksia Excelissä ja piirtää tämän jälkeen samoista elementeistä kaavio Visiolla. Käyttäjille tarjotaan täten mahdollisuus valita kuhunkin tilanteeseen parhaiten sopiva työkalu, ja he saavat itse päättää millä tavalla työskentelevät.

Excel-työkalun toteutus on melko geneerinen, ja siihen on mahdollista lisätä uusia näkymätyyppejä ilman koodiin tehtäviä muutoksia. Näkymätyyppeihin voi luoda template-elementtejä, jotka määrittävät niitä käyttävien elementtien ulkoasun sekä mallielementin tyypin. Tämän avulla työkalun käyttökohteita saadaan lisättyä esimerkiksi tekemällä näkymätyyppejä eri kohderyhmien tarpeita varten, joissa esitetään erilaisia tietoja samoista elementeistä. Lisäksi työkaluun saadaan tehtyä ympäristöön myöhemmin lisättäville mallinnuskielille Excel-näkymätyypit.

Suurin osa työkalun vaatimuksista saatiin toteutettua. Sen käytettävyydessä sekä suorituskyvyssä on kuitenkin vielä parannettavaa ja niitä tullaan kehittämään, sekä puuttuvat ominaisuudet tullaan toteuttamaan lähitulevaisuudessa. Tämän jälkeen työkalu annetaan koekäyttöön Trinityn teollisuusyhteistyökumppanille. Työkaluun tullaan myös todennäköisesti lisäämään tulevaisuudessa uusia toimintoja ja näkymätyyppejä.

LÄHTEET

[1] Koskimies, K., Koskinen, J., Maunumaa, M., Peltonen, J., Selonen, P., Siikarla, M. & Systa, T. UML työvälineenä ja tutkimuskohteena. Tietojenkäsittelytiede. 2004. ss. 19-51.

[2] Booch, G., Rumbaugh, J. & Jacobson, I. The Unified Modeling User Guide. Reading, Massachusetts: Addison Wesley; 1998. 512 p.

[3] van Gigch, J.P. System Design Modeling and Metamodeling. New York: Plenum Press; 1991. 461 p.

[4] Object Management Group. UML 2.3 Infrastructure Specification. [WWW]. [viitattu 9.4.2012]. Saatavissa: http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/.

[5] Object Management Group. Unified Modeling Language. [WWW]. [viitattu 9.4.20121]. Saatavissa: http://www.uml.org/.

[6] Object Management Group. UML 2.3 Superstructure Specification. [WWW]. [viitattu 9.4.2012]. Saatavissa: http://www.omg.org/spec/UML/2.3/Superstructure/PDF/.

[7] Object Management Group. OMG's MetaObject Facility. [WWW]. [viitattu 9.4.2012]. Saatavissa: http://www.omg.org/mof/.

[8] Microsoft. Excel 2010:n perustoiminnot. [WWW]. [viitattu 4.5.2012]. Saatavissa: http://office.microsoft.com/fi-fi/excel-help/excel-2010-n-perustoiminnot-HA101829993.aspx.

[9] Montalbano, E. Forrester: Microsoft Office in No Danger From Competitors. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://www.pcworld.com/businesscenter/article/166123/forrester_microsoft_office_in_no_danger_from_competitors.html?tk=nl_dnx_h_crawl.

[10] Microsoft. Excel Object Model Overview. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/wss56bz7%28v=vs.100%29.aspx.

[11] Microsoft. Microsoft.NET Framework. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://www.microsoft.com/net/.

[12] Getz, K. Understanding the Excel Object Model from a.NET Developer's Perspective. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa168292%28office.11%29.aspx.

[13] Microsoft. Properties (C# Programming Guide). [WWW]. [viitattu 5.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/x9fsa0sw%28v=vs.100%29.aspx.

[14] Microsoft. Visual C# Language. [WWW]. [viitattu 5.4.20112]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa287558%28v=vs.71%29.aspx.

[15] Microsoft. Excel Object Model Reference. [WWW]. [viitattu 5.4.2102]. Saatavissa: http://msdn.microsoft.com/en-us/library/bb149081%28v=office.12%29.aspx.

[16] Microsoft. Visual Studio Tools for Office. [WWW]. [viitattu 6.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/d2tx7z6d%28v=VS.80%29.aspx.

[17] Microsoft. Host Items and Host Controls Overview. [WWW]. [viitattu 9.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/9z4e3456%28v=vs.100%29.aspx.

[18] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. Pattern-Oriented Software Architecture, Volume 1, A System of Patterns. Chichester: Wiley; 1996. 476 p.

[19] Kosikimies, K. & Mikkonen, T. Ohjelmistoarkkitehtuurit. Jyväskylä: Talentum; 2005. 250 s.

[20] Gamma, E., Helm, R., Johnson, R. & Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Westford, Massachusetts: Addison-Wesley; 1995. 395 p.

[21] Microsoft. Exploring the Observer Design Pattern. [WWW]. [viitattu 17.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ee817669.aspx.

[22] Sybase. PowerDesigner. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://www.sybase.com/products/modelingdevelopment/powerdesigner.

[23] Peltonen, J. & Vartiala, M. An agent based architecture style for application integration. In: Sect. Comp., Volume 31; 2009. pp. 3-22.

[24] Muhametsin, S., Vartiala, M. & Peltonen, J. A Model for Language-Independent Mobile Agents. 12th Symposium on Programming Languages and Software Tools. 2011.

[25] Halme, K. Mallinnustyökaluympäristön hallintatyökalu. Tampere: Tampereen teknillinen yliopisto; 2011. 53 s.

[26] Peltonen, J., Felin, M. & Vartiala. M. From a freeform graphics tool to a repository based modeling tool. Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. 2010;ECSA '10: pp. 277-284.

[27] Felin, M. Microsoft Vision laajentaminen joustavaksi tietokantapohjaiseksi mallinnustyökaluksi. Tampere: Tampereen teknillinen yliopisto; 2011. 60 s.

[28] Haikala, I & Märijärvi, J. Ohjelmistotuotanto. 9. painos. Helsinki: Talentun Media Oy; 2003. 430 s.

[29] No Magic, Inc. Magic Draw. [WWW]. [viitattu 6.4.2012]. Saatavissa: https://www.magicdraw.com/.

[30] Grönniger, H., Krahn, H., Rumpe, B., Schindler, M. & Völkel, S. Text-based Modeling. Braunschweig, Germany: Institute of Software Systems Engineering.

[31] Microsoft. Use the Ribbon instead of toolbars and menus. [WWW]. [viitattu 5.4.2012]. Saatavissa: http://office.microsoft.com/en-us/help/use-the-ribbon-instead-of-toolbars-and-menus-HA010089895.aspx.

[32] Microsoft. Custom Task Panes Overview. [WWW]. [viitattu 6.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa942864%28v=vs.100%29.aspx.

[33] Microsoft. Programming Application-Level Add-Ins. [WWW]. [viitattu 10.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/bb157876.aspx.

LIITE 1: TRINITYN METAMETAMALLI