CoMa - Suunnitteludokumentti

Mindmap - Ilari Nieminen

Helsinki 16.12.2005

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Antti Kavonen
Ilari Nieminen
Tiina Torvinen
Valtteri Tyrsky
Kari Velling
Antti Vähäkotamäki

Asiakas

Anni Rytkönen

Johtoryhmä

Juho Teuho (ohjaaja)
Juha Taina

Kotisivu

`http://www.cs.helsinki.fi/group/mindmap`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset

Sisältö

1 Merkinnöistä

Tässä luvussa määritellään suunnitteludokumentissa käytetyt esitys- ja merkintätavat.

1.1 Luokkien kuvaus

Komponenttien luokkarakenne kuvataan luokkakaavioilla, ja tärkeimmät luokat kuvataan lisäksi sanallisesti seuraavassa aliluvussa esitetyllä tavalla. Luokasta kuvataan ainoastaan epätriviaalit kohdat, ja niinpä kaikkia seuraavassa esitettyjä kohtia ei kuvata jokaisen luokan osalta.

1.1.1 Luokan muodollinen esitystapa

Sanallinen kuvaus siitä, mikä tehtävä luokalla on järjestelmässä ja miten luokka sen hoitaa.

Luokan olioiden tilaan liittyvät rajoitteet

Sanallinen kuvaus niistä vaatimuksista ja rajoitteista, jotka ohjelma asettaa luokan instansseille. Esimerkiksi kenttien sallitut arvoalueet.

Luokkakohtaiset kentät

- *kentän 1 nimi : kentän 1 tyyppi*. Kentän 1 kuvaus
- *kentän 2 nimi : kentän 2 tyyppi*. Kentän 2 kuvaus
- *kentän n nimi : kentän n tyyppi*. Kentän n kuvaus

Oliokohtaiset kentät

- *kentän 1 nimi : kentän 1 tyyppi*. Kentän 1 kuvaus
- *kentän 2 nimi : kentän 2 tyyppi*. Kentän 2 kuvaus
- *kentän n nimi : kentän n tyyppi*. Kentän n kuvaus

Luokkakohtaiset metodit

- *java-syntaksin mukainen metodin 1 esittely*. Kuvaus metodin 1 toiminnasta
- *java-syntaksin mukainen metodin 2 esittely*. Kuvaus metodin 2 toiminnasta
- *java-syntaksin mukainen metodin n esittely*. Kuvaus metodin n toiminnasta

Konstruktorit

- *java-syntaksin mukainen konstruktorin 1 esittely*. Kuvaus konstruktorin 1 toiminnasta

- *java-syntaksin mukainen konstruktorin 2 esittely*. Kuvaus konstruktorin 2 toiminnasta
- *java-syntaksin mukainen konstruktorin n esittely*. Kuvaus konstruktorin n toiminnasta

Oliokohtaiset metodit

- *java-syntaksin mukainen metodin 1 esittely*. Kuvaus metodin 1 toiminnasta
- *java-syntaksin mukainen metodin 2 esittely*. Kuvaus metodin 2 toiminnasta
- *java-syntaksin mukainen metodin n esittely*. Kuvaus metodin n toiminnasta

1.1.2 Esimerkkiluokka: *class Selkärankainen extends Eläin*

Selkärankainen on eläin, jolla on selkäranka.

Luokan olioiden tilaan liittyvät rajoitteet

Selkärankainen-oliolla on oltava ei-null selkäranka.

Luokkakohtaiset kentät

- *selkärankaisLaskuri : int*. Pitää yllä syntyneiden selkärankaisten kokonaismäärää.

Oliokohtaiset kentät

- *selkäranka : Selkäranka*. Olion selkäranka.

Luokkakohtaiset metodit

- *public void kasvataSelkärankaislaskuria()*. Kasvattaa selkärankaislaskuri-muuttujaa yhdellä.

Konstruktorit

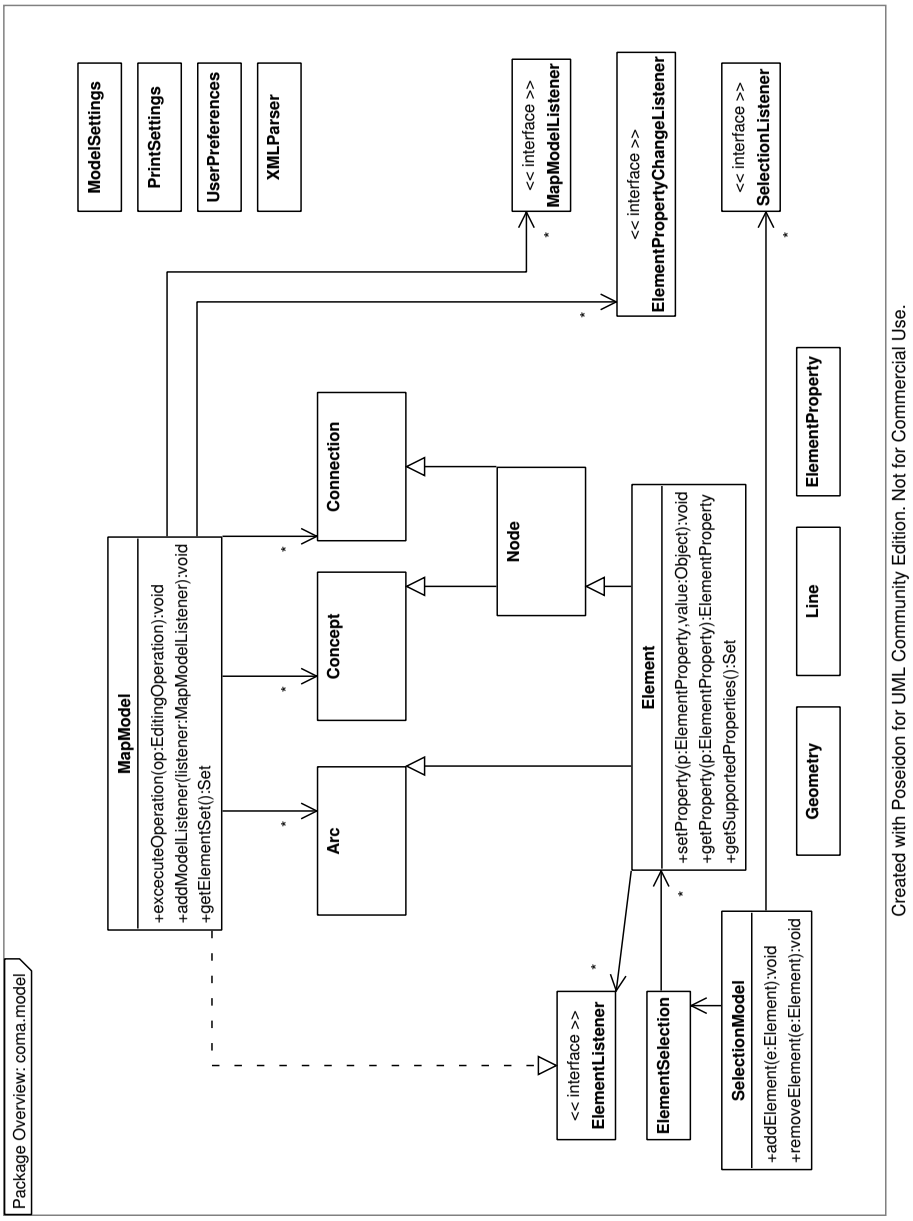
- *public Selkärankainen(Selkäranka s)*. Luo uuden Selkärankaisen eläimen annetulla selkärangalla sekä kasvattaa selkärankaislaskuria yhdellä.

Oliokohtaiset metodit

- *public int nikamia()*. Palauttaa nikamien määrän olion selkärangassa.

2 Järjestelmäarkkitehtuuri

CoMa järjestelmä jakautuu viiteen komponenttiin, joita kutakin vastaa yksi java pakkaus. Syksyn 2005 ohjelmistotuotantoprojektissa jätämme näistä toteuttamatta coma.server pakkauksen, sekä Moodle integraatiossa tarvittavan java- ja php-koodin.



Kuva 1: Model-komponentin luokkakaavio

2.1 Integraatio Moodle-oppimisympäristöön

CoMa ohjelma liitetään tulevaisuudessa mahdollisesti osaksi Moodle-oppimisympäristöä. Tämän projektin yhteydessä toteutamme Appletissa toimivan, miellekarttaeditorin. Editori toteutetaan itsenäisenä, toimivana ja käytettökelpoisena ohjelmana, joka on mahdollista kohtuullisella työmäärällä integroida Moodleen tai vastaavaan palvelinympäristöön.

Kaikki Editorin muokkaaman kartan sisällölliset muutokset toteutetaan käytännössä ainoastaan tekstimuotoista dataa sisältäviä EditingOperation olioita käyttämällä, joka mahdollistaa operaatioiden yksinkertaisen siirron verkon yli ohjelman toiselle instanssille.

Tähän tietorakenteeseen toteutetaan myös mahdollisuus palata mihin tahansa aikaisempaan tilaan (normaalin undo-pinon kadottamat haarat mukaanlukien), joka mahdollistaa

(muistinkäytön rajoissa rajattoman redo- ja undo- toiminnallisuuden lisäksi) yksinkertaisen palvelimen käytön usean käyttäjän samanaikaisessa editoinnissa.

Palvelinta ei toteuteta, mutta toteutettuaan se voi välittää järjestyksessä saamansa muutokset kaikille asiakkaille, joiden perusteella jokainen asiakas pystyy päättelemään mitkä hänen jo suorittamistaan operaatioista ovat ristiriidassa palvelimen aikaisemmin saamien muutosten kanssa ja kykenee kumoamaan nämä muutokset.

Suunniteltu asiakkaan toiminta-algoritmi noudattelee seuraavaa:

2.1.1 Muutosten teko

1. Suorita muutostoimenpide
2. Lisää muutos muutosrakenteeseen odottamaan hyväksymistä
3. Lähetä muutostoimenpide palvelimelle

2.1.2 Muutosten vastaanotto

1. Vastaanota muutostoimenpide
2. Jos muutostoimenpide on asiakkaan itse lähettämä ja muutostoimenpide on seuraava hyväksymätön operaatio muutostietorakenteessa, tallenna rakenteeseen tieto muutostoimenpiteen hyväksymisestä.
3. Jos muutostoimenpide on muun asiakkaan, kelaa takaisin kaikki hyväksymättömät muutostoimenpiteet ja suorita muutostoimenpide jos se on mahdollinen. Yritä tämän jälkeen suorittaa takaisin kelatut muutostoimenpiteet uudelleen. Ilmoita käyttäjälle jos toisen käyttäjän muutos on johtanut tilaan, jossa takaisin kelatut muutokset eivät ole suoritettavissa (konfliktitilanne).

Tällä tavalla toteutettuna palvelimen ei tarvitse olla tietoinen kartan rakenteesta ja se voidaan helposti toteuttaa Moodle-alustalle.

2.2 Model-View-Controller suunnittelumalli

Komponentit coma.model, coma.view ja coma.control toteutetaan pääsääntöisesti Model-View-Controller suunnittelumallin mukaisesti (jatkossa MVC). Komponentit esitellään yksityiskohtaisesti seuraavissa luvuissa.

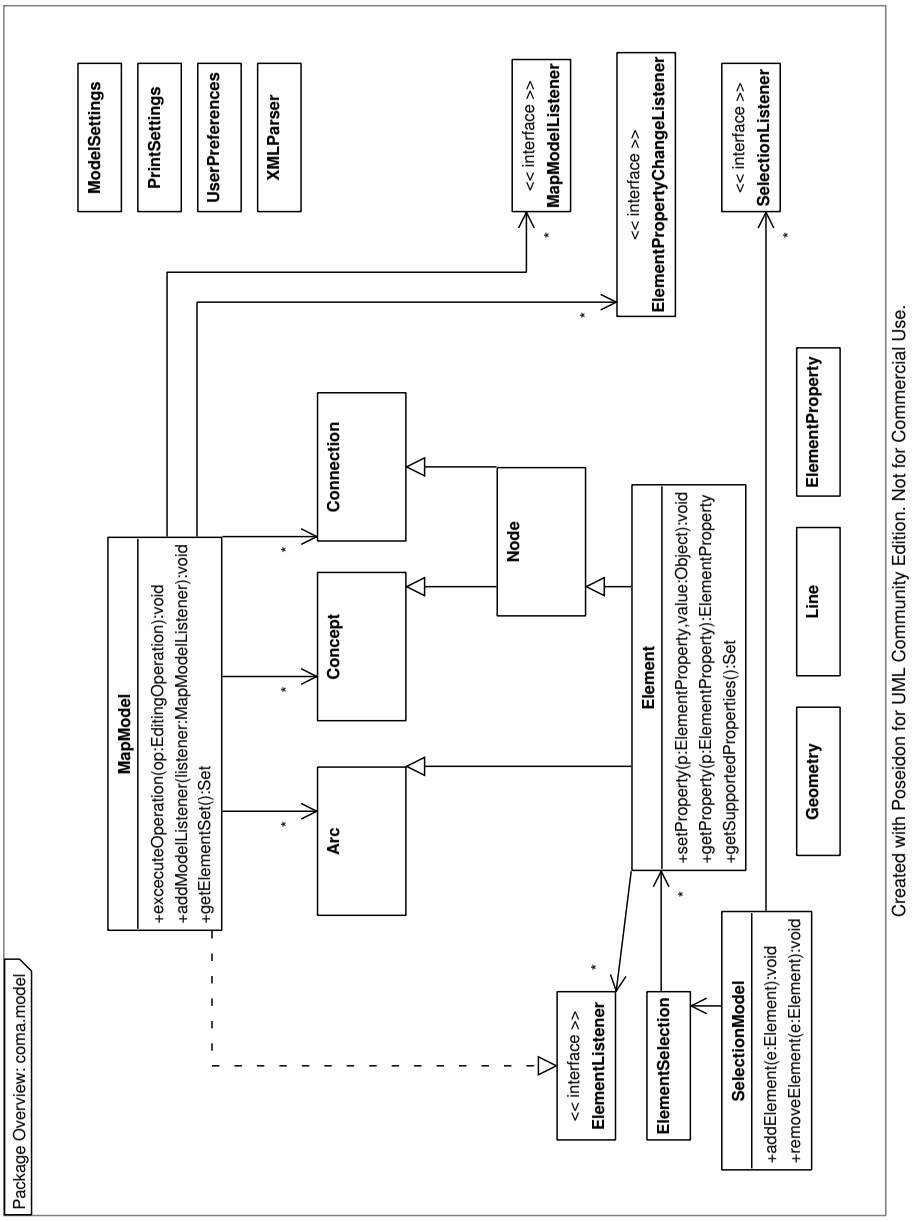
Model-komponentti käsittää miellekartan mallin ja tietosisällön, sekä miellekartan muokkaamisessa ja tallentamisessa tarvittavat tietorakenteet ja työkalut. MVC:n mukaisesti komponentissa ei määritellä miellekartan piirtorutiineita, vaan kaikki miellekartan visualisointia suorittava koodi on view-komponentissa.

View-komponentti käsittää java Appletin, sekä tähän kytketyt komponentit, joiden avulla käyttäjälle visualisoidaan miellekartan tila. *View*-komponentissa määritellään myös valikot sekä muut kontrollit joilla käyttäjä voi muokata miellekarttaa. *View*-komponentti

kuuntelee muutoksia miellekartassa ja pitää kartan visualisoinnin ajan tasalla muutosten sattuessa.

Controller-komponentti kuuntelee käyttäjältä saatua syötettä ja teettää käyttäjän pyytämät muutokset *Model*-komponentin miellekartassa. Ohjelman toimintalogiikka määritellään *Controller*-komponentin MainController luokassa.

3 Model-komponentti



Kuva 2: Model-komponentin luokkakaavio

3.1 Komponentin tarjoama palvelurajapinta

3.1.1 Miellekartan luominen

MapModel luokan instanssit ovat itsenäisiä miellekarttoja. Tyhjän miellekartan voi luoda MapModel() konstruktorilla. Mikäli miellekartan muutoksia halutaan kuunnella, addMapModelListener() metodilla voi lisätä MapModelListener-rajapinnan toteuttavan kuuntelijan kuuntelemaan haluttua miellekarttaa. Kaikista miellekarttaan kohdistuvista muutoksista ilmoitetaan kytketyille kuuntelijoille MapModelListener-rajapinnan metodien avulla.

3.1.2 Miellekartan muokkaaminen

MapModel instanssia, eli miellekarttaa muokataan executeOperation(op) -metodin kautta. Parametrina välitetty operaatio kuvailee muutokset joita karttaan halutaan tehdä (katso Control-komponentin EditingOperation-luokka).

3.1.3 Miellekartan tallentaminen ja lataaminen

XMLParser luokka tarjoaa tarvittavat työkalut miellekarttojen tallentamiseksi ja lataamiseksi tiedostosta. XMLParser.writeFile(...) tallentaa kartan levylle ja XMLParser.readFile(...) lukee kartan levyltä. Sekä lukeminen että kirjoittaminen aiheuttaa poikkeuksen, mikäli tiedosto on korruptoitunut tai järjestelmässä tapahtuu IO-virhe (esim. kovalevyllä ei ole tilaa).

3.1.4 Miellekartan ja käyttäjän asetusten muokkaaminen

UserPreferences-luokka sisältää käyttäjän mieltymysasetukset, joita käyttäjä voi muuttaa ohjelman työkalupalkista. Asetusten arvoja voi muuttaa setPreference()- metodilla ja kysellä getPreference()-metodilla.

ModelSettings-luokka määrittelee oletusasetukset miellekartan käsitteille ja yhteyksille. Näitä käytetään luotaessa uusia elementtejä karttaan. Oletusasetuksia voi muokata setDefaultProperty(Class, Object)-metodilla ja kysellä getDefaultProperty(Class)-metodilla.

3.1.5 Elementtien valitseminen

SelectionModel luokka tarjoaa tietorakenteen elementtijoukon käsittelemiselle. elementti on CoMa-ohjelmassa *valittu* jos se kuuluu SelectionModeliin. Elementtejä voi lisätä ja poistaa valituista elementeistä addElement(Element e)-, addElements(Set<Element> e)-setSelection(Element e)-, setSelection(Set<Element> e)-, removeElement(Element e) sekä clearSelection()- metodeilla.

Mikäli SelectionModelin muutoksia halutaan kuunnella, addSelectionListener()-metodilla voi lisätä SelectionListener-rajapinnan toteuttavan olion kuuntelemaan muutoksia halutussa SelectionModelissa.

3.2 Tärkeimmät luokat

3.2.1 class MapModel

MapModel on käsitekartan malli. MapModel-olio on verkko, jonka solmut ovat Node-olioita ja kaaret Arc-olioita. Node-oliot ovat joko Concept- tai Connection-tyyppisiä.

MapModel-luokan metodit määrittelevät rajapinnan, joilla muut järjestelmän komponentit voivat kommunikoida mallin kanssa. MapModel-olion tilassa tapahtuvia muutoksia kuunnellaan lisäämällä MapModelListener-rajapinnan toteuttava olio addMapModelListener()-metodilla.

Mallin tilaa muutetaan julkisen executeOperation(EditinOperation op) kautta.

MapModel-olioiden tilaan liittyvät rajoitteet

MapModel-olion edustamassa verkossa ei saa olla kaaria, jotka viittavat sellaisiin solmuihin, joita verkossa ei ole.

Oliokohtaiset kentät

- *nodes : java.util.Set<Node>*. Käsitekartan solmut, eli käsitteet ja yhteydet.
- *arcs : java.util.Set<Arcs>*. Käsitekartan kaaret, eli solmujen väliset viivat.
- *listeners : java.util.Set<MapModelListener>*. Käsitekartan kuuntelijat.
- *id : int*. Käsitekartan identifioiva luku.

Konstruktorit

- *public MapModel()*. Luo uuden tyhjän käsitekartan. Konstruktori alustaa kaikki listat tyhjiksi.
- *public Model(int id)*. Luo tyhjän kartan annetulla identifioijaluvulla.

Oliokohtaiset metodit

- *public java.util.Vector<Element> executeOperation(control.EditingOperation op)*. Suorittaa annetun operaation mallissa. Operaatio voi kuvailla yhden tai useamman elementin lisäyksen, poistamisen tai muokkaamisen. Operaatio joko onnistuu kokonaisuudessaan, tai muuten malliin ei tehdä mitään muutoksia. Metodi aiheuttaa IllegalModelOperation -poikkeuksen, mikäli operaatiota ei voida suorittaa (esimerkiksi sellaisen elementin poistaminen jota mallissa ei ole).

  Palauttaa listan kaikista mallin elementeistä, joiden ominaisuuksiin operaatio vaikutti, eli listan niistä elementeistä, jotka täytyy piirtää uudelleen operaation jälkeen.

  Kaikista malliin kohdistuvista muutoksista ilmoitetaan rekisteröidyille MapModelListener-olioille.

- *public void addMapModelListener(MapModelListener l)*. Lisää annetun kuuntelijan *listeners*-listaan.

- *public java.util.Set<Element> getElementSet()*. Palauttaa joukon, joka sisältää kaikki mallin elementit.
- *public java.util.Iterator<Node> iterateNodes()*. Palauttaa iteraattorin, joka käy läpi kaikki mallin solmut mielivaltaisessa järjestyksessä. Palautettu iteraattori on read-only tyyppinen, jotta sen kautta ei voi muokata mallin rakennetta.
- *public java.util.Iterator<Arc> iterateArcs()*. Palauttaa iteraattorin, joka käy läpi kaikki mallin kaaret mielivaltaisessa järjestyksessä. Palautettu iteraattori on read-only tyyppinen, jotta sen kautta ei voi muokata mallin rakennetta.

3.2.2 abstract class Element

Element on yhteinen yliluokka käsitekartan käsitteille, yhteyksille sekä näitä yhdistäville kaarille. Element koostuu joukosta ElementProperty- tyyppisiä ominaisuuksia, jotka määrittelevät esimerkiksi elementin sijainnin, värin ja muodon.

Elementissä tapahtuvia muutoksia voi kuunnella ElementListener- olion avulla. Kaikista elementin ominaisuuksien muutoksista ilmoitetaan kuuntelijoille. Element-luokan perivissä luokissa on määriteltävä getSupportedProperties()-joukko, joka kertoo mitä elementtien ominaisuuksia perivä luokka käyttää.

ElementProperty enumeraatio luettelee kaikki mahdolliset elementin ominaisuudet.

Oliokohtaiset kentät

- *properties : java.util.Map<ElementProperty, Object>*. Elementin ominaisuudet.
- *listeners : java.util.List<ElementListener>*. Elementin kuuntelijat, joille ilmoitetaan muutoksista ominaisuuksissa.

Konstruktorit

- *protected Element(int id, Map<ElementProperty, Object> properties)*. Luo uuden elementin annetuilla ominaisuuksilla. Mikäli jokin elementin käyttämistä ominaisuuksista ei sisälly annettuun joukkoon, kyseinen ominaisuus alustetaan oletusarvoiseksi. Näin saavutetaan yhteensopivuus tulevien ohjelmaversioiden kanssa: mikäli uusi ominaisuustyyppi lisätään ohjelmaan, tiedostosta voi yhä ladata vanhan tyyppisen elementin.

Oliokohtaiset metodit

- *public static Element constructElementFromStep(MapModel model, control.EditingOperationStep step)* . Luo elementin annetun muokkausoperaation perusteella. Luotu elementti on tyyppiä Arc, Concept tai Connection. Aiheuttaa poikkeuksen, mikäli annettu muokkausoperaation askel ei kuvaile elementtiä.

Oliokohtaiset metodit

- *public void setProperty(ElementProperty p, Object value)*. Asettaa elementille ominaisuuden p arvoksi annetun olion. Aiheuttaa poikkeuksen mikäli perivä Element-aliluokka ei tue ominaisuutta p. Kuuntelijoille ilmoitetaan muutoksesta.
- *public Object getProperty(ElementProperty p)*. Palauttaa ominaisuuden p arvon. Palauttaa null, mikäli arvoa ei ole asetettu.
- *public boolean isPropertySet(ElementProperty p)*. Palauttaa true joss ominaisuus p ei ole elementillä null.
- *public boolean isSupportedProperty(ElementProperty p)*. Palauttaa true joss perivä aliluokka tukee ominaisuutta p.
- *public abstract java.util.Set<ElementProperty> getSupportedProperties()*. Palauttaa joukon, joka sisältää perivän aliluokan tukemat ominaisuudet. Korvattava perivissä aliluokissa.

3.2.3 abstract class Node extends Element

Node toimii yliluokkana käsitekartan käsitteille ja yhteyksille. Käsitekartta on verkko, jonka solmut ovat Node-olioita. Node-olio ei ole tietoinen käsitekartan rakenteesta, erityisesti omista naapurisolmuistaan. Sen sijaan kaikki verkon rakenteeseen kohdistuvat operaatiot toteutetaan *Model*-luokassa.

3.2.4 class Concept extends Node

Concept-olio vastaa yhtä käsitekartan käsitettä. Käsite visualisoidaan laatikoksi tai vastaavaksi alueeksi, jonka sisällä on tekstiä.

3.2.5 class Connection extends Node

Connection-olio vastaa käsitekartan kahden käsitteen välistä yhteyttä. Teknisesti Connection on käsitekarttaa vastaavan verkon solmu, mutta ohjelman käyttöliittymässä Connection esittää Kahden tai useamman käsitteen välistä kaarta.

3.2.6 class Arc extends Node

Arc-olio vastaa käsitekartalla kahden elementin välistä viivaa, eli käsitekartan muodostaman verkon kahden solmun välistä kaarta. CoMa ohjelmassa käsitekartan rakenne takaa, että kaari yhdistää aina yhden Concept olion yhteen Connection olioon.

Oliokohtaiset kentät

- *connection : Connection*. Yhteyssolmu, joka määrittää kaaren toisen päätepisteen.
- *concept : Concept*. Käsitesolmu, joka määrittää kaaren toisen päätepisteen.
- *model : MapModel*. Käsitekartta, jossa kaari sijaitsee.

3.2.7 class Geometry

Geometry määrittelee elementin fyysisen muodon. Muoto esitetään java.awt.geom pakkauksen Shape ja Area-olioita käyttäen. Muoto koostuu 'rungosta' (body), sekä 'kontrollimuodoista' (controlShapes).

Runko on elementin pääasiallinen muoto, joka ei riipu käyttäjän preferensseistä, kuten painikkeiden näkyvyydestä. Kontrollimuodot ovat väliaikaisia ulokkeita rungon ohella, kuten yhteyspainike tai kommenttikuvake. Elementin alue on yhdiste rungosta ja kontrollimuodoista.

Geometry-olioita käytetään elementtien renderoimiseen View-komponentissa ja niinpä Geometry olio ei sijaitse elementin sisällä, vaan elementtiin liitetyssä ElementRenderer-oliossa. Geometry-olioita tarvitaan kuitenkin MapModel-luokassa viivojen sijoittelualgoritmeissa, joten Geometry-luokka on määritelty Model-komponentissa.

Geometry-olio ei sisällä mitään uniikkia informaatiota elementistä, vaan kaikki geometrian laskemiseksi tarvittava tieto löytyy elementin ominaisuuksista (esim, $SHAPE, LOCATION, FONT_SIZ$ ElementRenderer oliot kuuntelevat muutoksia elementeissä ja päivittävät Geometry-oliota tarvittaessa.

Konstruktorit

- *public Geometry(Shape body)*. Muodostaa uuden geometrian annetulla rungolla, ilman kontrollimuotoja.

Oliokohtaiset metodit

- *private void validateArea()*. Asetaa area-muuttujan arvoksi yhdisteen body U controlShapes. Tätä metodia kutsutaan aina kun body muuttuu, controlShape lisätään tai controlShape poistetaan.
- *public ControlID getControlAt(P point)*. Palauttaa annetusta pisteestä löytyvän kontrollimuodon tyypin. Palauttaa null, mikäli annetussa pisteessä ei ole kontrollimuotoa.
- *public boolean containsPoint(Point p)*. Palauttaa *true* joss area sisältää pisteen p, eli body tai jokin controlShape-olio sisältää pisteen p. Muutoin palauttaa *false*.

3.2.8 class ModelSettings

ModelSettings-olio sisältää Model-komponentin käyttäjäasetukset. Näitä ovat oletusarvoinen käsitteen tyyli, oletusarvoinen yhteyden tyyli ja oletusarvoinen kaaren tyyli.

Konstruktorit

- *public ModelSettings()*. Luo oletusarvoiset mallin asetukset.

Oliokohtaiset metodit

- *public Object getDefaultProperty(Class c, ElementProperty p).* Palauttaa luokan c (Concept, Connection tai Arc) oletusarvon ominaisuudelle p. Palauttaa null, mikäli oletusarvoa ei ole asetettu.
- *public void setDefaultProperty(Class c, ElementProperty p, Object value).* Asettaa luokan c (Concept, Connection tai Arc) oletusarvon ominaisuudelle p.

4 Controller-komponentti

Controller-komponentti vastaa ohjelman kulun ohjaamisesta, käyttäjän syötteiden käsittelemisestä sekä miellekartan undo-pinon säilyttämisestä ja hallinnoinnista.

Ohjelman kulun ohjaaminen toteutetaan tilasiirtymäpohjaisesti (katso tilasiirtymäkaavio). Ohjelman mahdolliset tilat numeroidaan ProgramState-enumeraatiossa. MainController-luokka toimii ohjelman "selkärankana"ja toteuttaa käyttäjän syötteiden käsittelemisen sekä tarvittavat toimenpiteet, jotka käyttäjän syötteet aiheuttavat käsiteltävänä olevalle miellekartalle.

Miellekartan undo-pino toteutetaan listarakenteena, joka sisältää tiedot kaikista käyttäjän tekemistä muutoksista miellekarttaan. Undo-pinon kokoa ei rajata.

4.1 Komponentin tarjoama palvelurajapinta

4.1.1 Ohjelman kontrollitapahtumien kuunteleminen

ControlListener-rajapinta määrittelee kontrollitapahtumakuuntelijan rakenteen. MainController-olioon voi liittää ControlListener-kuuntelijan tarkkailemaan kontrollitapahtumia.

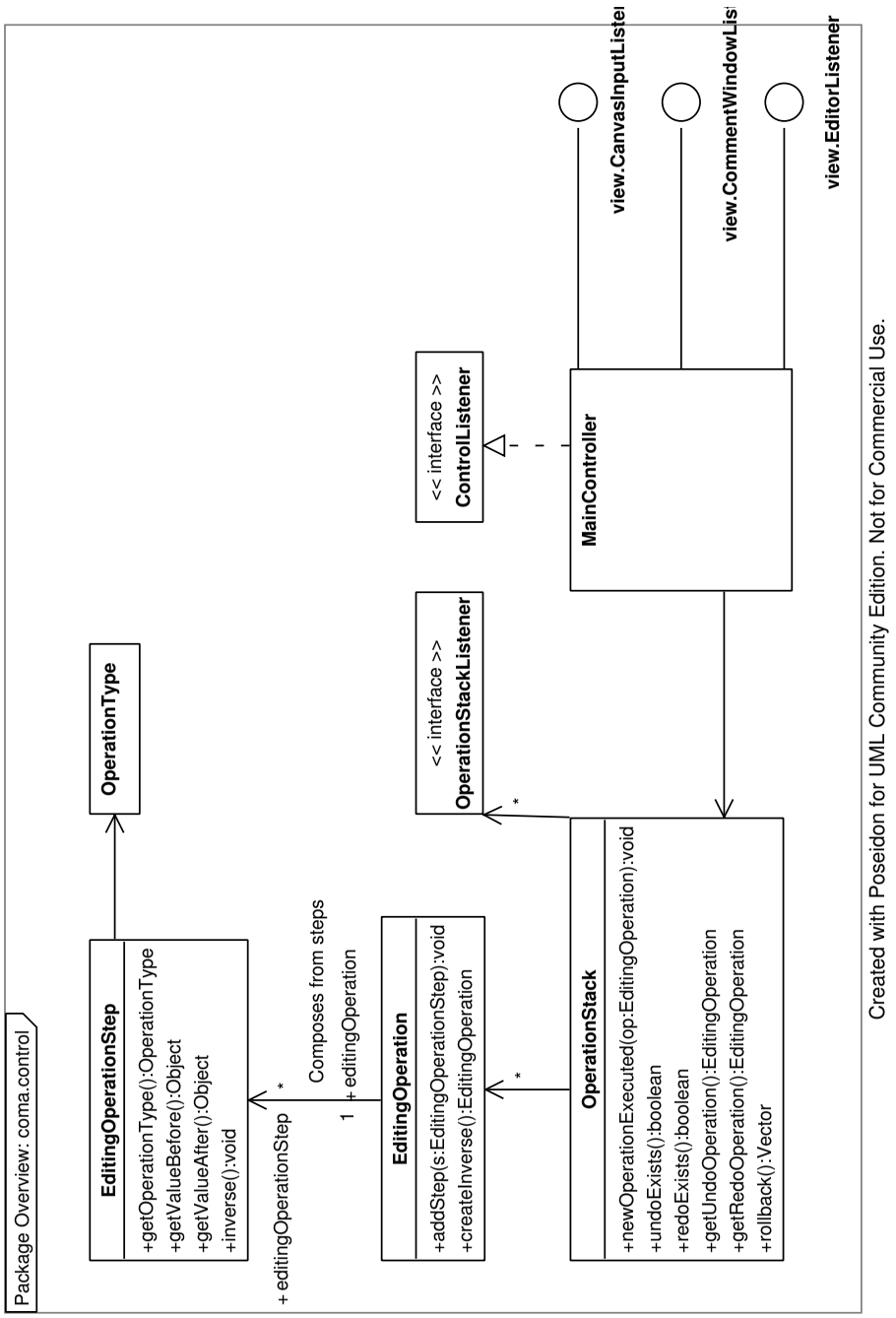
ControlType-enumeraatiossa on lueteltu kaikki ohjelmassa esiintyvät kontrollitapahtumat. Kaikille MainController -luokkaan liitetyille kuuntelijoille ilmoitetaan uusista kontrollitapahtumista niiden sattuessa.

4.1.2 Kontrollitapahtumien laukaiseminen

Ohjelman kontrollitapahtumia voi laukaista kutsumalla MainController-olion julkisia metodeita. Kontrollien laukaisumetodien nimet päättyvät -Issued()", esimerkiksi undoIssued() laukaisee kontrollitapahtuman, joka vastaa yhden undo-toimenpiteen suorittamista. Kaikista kontrollitapahtumista ilmoitetaan MainController-olioon liitetyille ControlListener-kuuntelijoille. Laukaisumetodit kuvataan tarkemmin MainController-luokan kuvauksen yhteydessä.

4.2 Komponentin tarvitsema palvelurajapinta

Controller-komponentti kytkeytyy sekä käsiteltävänä olevaan miellekarttaan (model.MapModel-olio), sekä tätä karttaa piirtävään graafiseen ympäristöön (view.Canvas-olio sekä view.AppletWindow-olio). MainController-olio sisältää viitteet kontrolloimiinsa miellekarttaan sekä graafiseen



Kuva 3: Controller-komponentin luokkakaavio

ympäristöön. Viitteiden kautta MainController teetättää käyttäjän pyytämät toimenpiteet MapModel-olion aksessorien kautta, sekä visualisoi käyttäjän tarvitsemia tietoja käyttäjälle Canvas-olion aksessorien kautta.

4.3 Tärkeimmät luokat

4.3.1 class MainController

MainController-luokka ohjaa ohjelman toimintaa. MainController-olio kerää käyttäjän syötteet ohjelmalle ja delegoi niiden perusteella muokkaustoimenpiteitä käsitekartan malliin (MapModel) sekä piirtämistoimenpiteitä käsitekartan piirtäjälle (Canvas).

Controller pitää myös yllä luetteloa kaikista käyttäjän suorittamista muokkaustoimenpiteistä ja tarjoaa mahdollisuuden peruuttaa operaatioita ja suorittaa peruutettuja operaatioita uudelleen.

MainController pitää yllä model.SelectionModel-rakennetta, joka vastaa käyttäjän valitsemien elementtien joukkoa. Suuri osa ohjelman kontrolleista muokkaa juuri valittujen elementtien joukkoa.

Julkiset kontrollien laukaisumetodit

- *public void createConceptIssued()*. Uuden, automaattisesti sijoitellun käsitteen luominen.
- *public void aboutIssued()*. About-ikkunan näyttäminen.
- *public void launchHelpIssued()*. Käyttöohjeen näyttäminen.
- *public void removeElementsIssued()*. Valittujen elementtien poistaminen.
- *public void setAsDefaultIssued()*. Valitun elementin tyylin asettaminen oletusarvoiseksi tyyliksi tuleville elementeille.
- *public void propertyChangeIssued(ElementProperty property, Object value, Class elementClass)*. Valittujen elementtien ominaisuuden muuttaminen.
- *public void toggleCommentIssued()*. Mikäli valitun elementin kommentti-ikkuna ei ole näkyvissä, saattaa tämän näkyviin, muutoin sulkee sen.
- *public void undoIssued()*. Viimeisimmän miellekarttaa koskevan muokkaustoimenpiteen peruuttaminen.
- *public void redoIssued()*. Viimeisimmän peruutustoimenpiteen kumoaminen.
- *public void newFileIssued()*. Uuden miellekartan avaaminen, sekä tarvittaessa nykyisen tallentaminen.
- *public void closeAllCommentsIssued()*. Kaikkien avoimien kommentti-ikkunoiden sulkeminen.

4.3.2 class OperationStack

OperationStack toteuttaa undo tapahtumia säilövän tietorakenteen. Operaatioita voi säilöä rakenteeseen

Oliokohtaiset metodit

- *public int newOperationExecuted(EditingOperation operation)*. Säilöö annetun operaation tietorakenteeseen. Seuraava undo-operaation peruuttaa tämän lisätyn operaation.
- *public boolean undoExists()*. Kertoo, onko rakenteesta saatavissa undo-operaatio.
- *public boolean redoExists()*. Kertoo, onko rakenteesta saatavissa redo-operaatio.
- *public EditingOperation getUndoOperation()*. Palauttaa operaation, joka peruuttaa viimeisimmän rakenteeseen lisätyn operaation, sekä päivittää OperationStack-rakennetta siten, että seuraava getUndoOperation() palauttaa viimeistä edellisen operaation jne...
- *public EditingOperation getRedoOperation()*. Palauttaa operaation, joka suorittaa uudelleen viimeksi perutun operaation, sekä päivittää OperationStack-rakennetta.

4.3.3 class EditingOperation

EditingOperation kuvaa yhden, atomisen muokkausoperaation joka voi koostua yhdestä tai useammasta askeleesta (EditingOperationStep). Operaation voi katenoida toisen operaation kanssa, jolloin näiden operaatioiden askeleet suoritetaan peräkkäin yhtenä, atomisena kokonaisuutena. Operaation voi myös kääntää, jolloin jokainen operaation askel käännetään ja askeleet suoritetaan käänteisessä järjestyksessä.

4.3.4 class EditingOperationStep

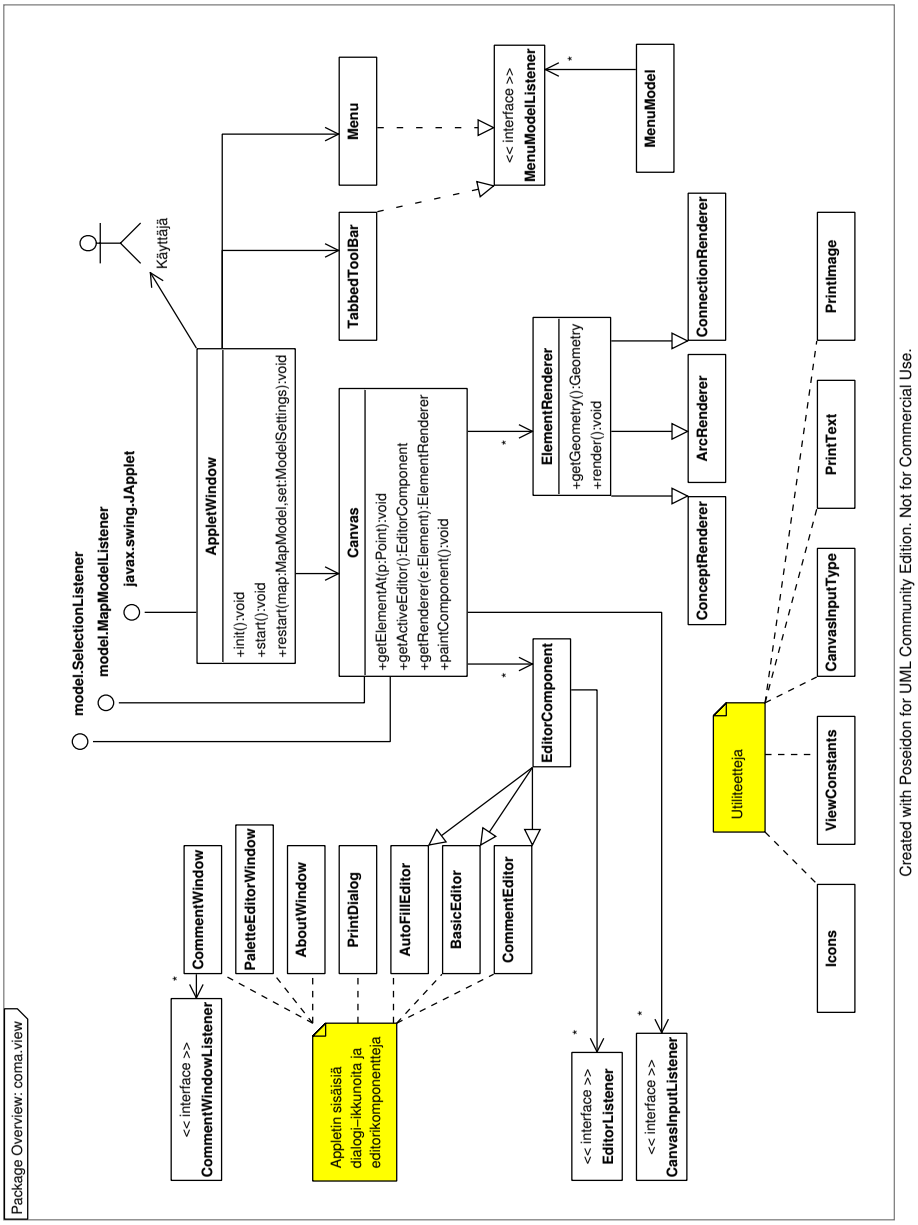
EditingOperationStep kuvaa yhden muokkaustoimenpideaskeleen. Askel sisältää identifioivan luvun, joka määrää operaation tyypin (esimerkiksi käsitekartan elementin poistaminen). Lisäksi askel sisältää kaksi Object-oliota, jotka kuvaavat operaation kohteena olevan olion tilan ennen operaatiota ja sen jälkeen. Operaation voi kääntää vaihtamalla ennen- ja jälkeen-olioiden paikkaa.

5 View-Komponentti

5.1 Komponentin tarjoama palvelurajapinta

Canvas-luokka on view-komponentin keskeisin luokka ja tarjoaa kustomoidun java-komponentin sekä MapModel-olion visualisointiin, sekä erinäisten dialogien näyttämiseen käyttäjälle. MapModel-olion elementeille luodaan oliokohtaiset renderöijät Canvas.getRenderer() metodilla. Dialogeja avataan Canvas-metodin julkisten aksessorien kautta. AppletWindow-luokka perii JApplet-luokan ja kytkee Canvas-olion swing-komponenttihierarkiaan.

Koska miellekartan elementtien geometria (muoto ja koko) liittyvät hyvin vahvasti elementtien piirtämiseen, elementtien sijaitsemista miellekartan eri kohdissa kysellään Canvas-oliolta eikä MapModel-oliolta.



Kuva 4: View-komponentin luokkakaavio

5.2 Komponentin tarvitsema palvelurajapinta

View-komponentti visualisoi MapModel-olion, joten view-komponentin tarvitsema palvelurajapinta on model-luokan tarjoama palvelurajapinta. Lisäksi View-komponentti kytkeytyy javan Swing arkkitehtuurin kautta ohjelmaa suorittavaan JApplet sovelmaan, sekä sitä kautta java virtuaalikoneeseen.

5.3 Tärkeimmät luokat

5.3.1 class Canvas

Canvas toteuttaa tarpeen mukaan rajattomasti kasvavan, sekä rajallisesti suppenevan piirtopinnan, jolla CoMa järjestelmässä visualisoidaan miellekartta. Canvas-luokka vastaa kaikesta MapModel-miellekartan elementtien piirtämisestä. Lisäksi Canvaksella näytetään erinäiset dialogit, esimerkiksi kommentti-ikkuna ja muut vastaavat tekstieditorit.

Canvas-luokka perii util.TransformablePanel-luokan, joka perii javax.swing.JPanel-luokan, joten Canvas on sellaisenaan itsenäinen ja toimiva java-komponentti, jonka voi liittää osaksi javan Swing-komponenttihierarkiaa.

Oliokohtaiset metodit

- *public void paintComponent(Graphics g, Rectangle clipRect, double scale)*. Piirtorutiini, jolla Canvas piirretään Swing-ympäristössä.
- *public void paintToImage(BufferedImage image)*. Piirtää käsiteltävänä olevan miellekartan annettuun kuvaan.
- *public void setControlsVisibleOnlyNearMouse(boolean b)* . Asettaa miellekartan elementteihin liittyvien painikkeiden näkyvyyspreferenssin: painikkeet näkyvät Canvaksella joko jatkuvasti, tai valinnan mukaan ainoastaan hiiren läheisyydessä.
- *public void showEditorComponent(EditorComponent editor, P location)* . Näyttää annetun editorikomponentin Canvaksella. Lähes kaikki käyttäjälle osoitetut editorit perivät EditorComponent luokan. Canvaksella voi olla enintään yksi editorikomponentti kerrallaan.
- *public void hideEditorComponent()* . Sulkee Canvaksella aktiivisen editorikomponentin, mikäli tällainen löytyy.
- *public ElementRenderer getRenderer(Element element)*. Luo ja palauttaa piirtäjäolion annetulle miellekartan elementille.
- *public Element getElementAt(P point)* . Palauttaa elementin annetusta pisteestä Canvakselta. Palauttaa null, mikäli pisteessä ei ole elementtiä.

5.3.2 class MenuModel

MenuModel toteuttaa loogisen valikkorakenteen ohjelman kontrolleille. CoMa-ohjelman työkalurivi ja valikkorivi toimivat käyttöliittymänä MenuModel-kontrolleille.

MenuModel sisältää jokaista valikosta käytettävää ohjelman kontrollia kohden oman Action-luokan, joka määrittelee kontrollin nimen, mahdollisen ikonin, ToolTip-tekstin sekä kontrollin seuraukset. Tyypillisesti action- kutsuu MainController-luokan kontrollinlaukaisumetodeita.

MenuModel toimii tietorakenteena, johon säilötään kontrolli-Action-olioita. MenuModelListener-oliot (TabbedToolBar ja Menu) saavat ilmoituksen aina, kun uusi Action lisätään MenuModel-olioon, ja lisäävät tarvittaessa Actionille sopivan käyttöliittymäkomponentin, jolla käyttäjä voi laukaista kontrollin.

5.3.3 abstract class EditorComponent extends JPanel

EditorComponent toimii yliluokkana kaikille CoMa-ohjelman interaktiivisille editoreille, kuten käsitteen nimieditorille (BasicEditor), yhteyden nimieditorille (AutoFillEditor) sekä kommenttieditorille (CommentWindow).

5.3.4 abstract class ElementRenderer implements ElementListener

ElementRenderer toimii yliluokkana miellekartan elementtien piirtäjäolioille (ConceptRenderer, ConnectionRenderer, ArcRenderer). CoMa-järjestelmässä jokaisella elementillä on oma piirtäjä, joka pitää yllä elementin geometriaan liittyviä laskelmia. ElementRenderer kuuntelee muutoksia piirtämässään elementissä ElementListener rajapinnan kautta ja täten kaikki muutoksen elementeissä aiheuttavat välittömästi myös kuvan päivittymisen.

5.3.5 Dialogi-ikkunaluokat

Seuraavat luokat toteuttavat erinäisiä Swing-yhteensopivia dialogi-ikkunoita, joita käyttäjälle näytetään CoMa-ohjelmassa. Ikkunat näytetään Canvas-komponentissa sisäisinä ikkunoina, tai Appletista ponnahdusikkunoina.

- *AppletWindow*. Perii JApplet-luokan. Toimii CoMa ohelman suoritusalustana.
- *AboutWindow*. Näyttää käyttäjälle CoMa-ohjelman tekijät sekä lisenssin.
- *PaletteEditorWindow*. Editori-ikkuna, jossa käyttäjä voi määrittää ohjelman käyttämän väripaletin.
- *PrintDialog*. Tulostusdialogi, jossa käyttäjä voi valita tulostusasetukset (kuvan asettelu ja kommenttien tulostuminen) sekä käytettävän tulostimen.

6 Järjestelmän tilasiirtymät

7 Sekvenssikaavioita

Ohessa on ohjelman joidenkin epätriviaalien toimintojen toteutukseen liittyviä sekvenssikaavioita. Kaavioissa tapahtumat etenevät ylhäältä alaspäin, ja vertikaaliset laatikot kuvaavat metodien suorittamista.

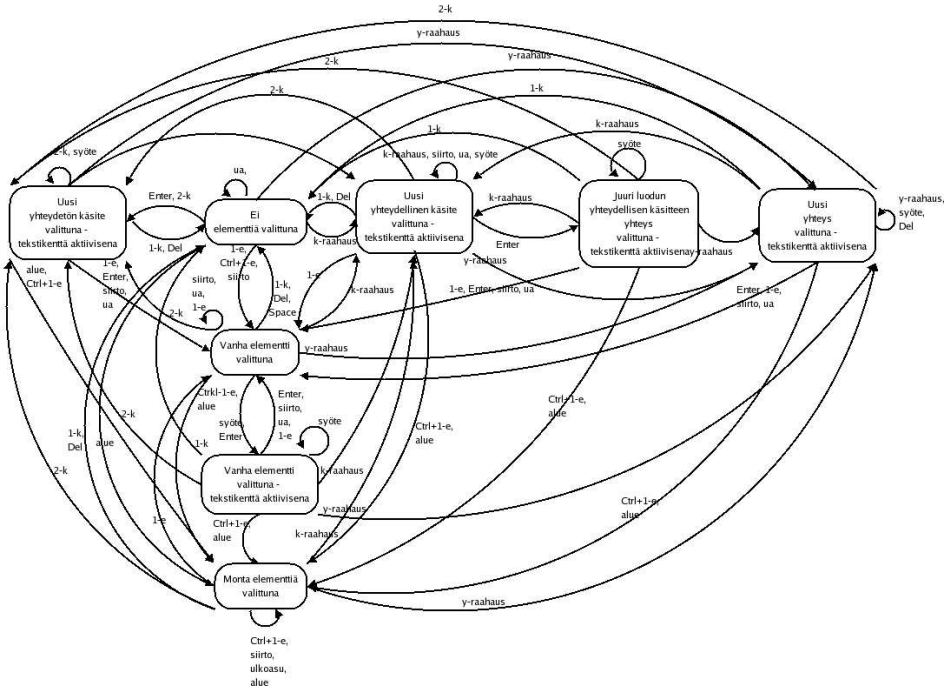
Tila	syöte	Enter	1-k	1-e	2-k	k-raahaus	y-raahaus	Del	siirto	ulkoasu l. ua	Ctrl + 1-e	alue	t
1 Uusi yhteydetön käsite valittuna - tekstikenttä aktiivisena	1	2	3	2	1*	4	5	1**	2	2	6	6	
2 Vanha elementti valittuna	7	7	3	2	1	4	5	3	2	2	6	6	
3 Ei elementtiä valittuna	-	1	-	2	1	4	5	-	-	-***	2	6	
4 Uusi yhteydellinen käsite valittuna - tekstikenttä aktiivisena	4	8	3	2	1	4*	5	4**	4	4	6	6	
5 Uusi yhteys valittuna - tekstikenttä aktiivisena	5	2	3	2	1	4	5*	5**	2	2	6	6	
6 Monta elementtiä valittuna	-	-	3	2	1	4	5	3	6	6	6****	6*****	
7 Vanha elementti valittuna - tekstikenttä aktiivisena	7	2	3	2	1	4	5	7**	2	2	6	6	
8 Juuri luodun yhteydellisen käsitteen yhteys valittuna - tekstikenttä aktiivisena	8	2	3	2	1	4	5	8**	2	2	6	6	

* Ei palata samaan tilaan, vaan luodaan uusi elementti, jolla on sama tila.
** osa syötettä
*** Muuttaa oletusarvoja
**** Ei palata samaan tilaan, vaan tilaan, jossa valittuna on yksi enemmän
***** Ei palata samaan tilaan, vaan tilaan, jossa uusi joukko on valittuna.

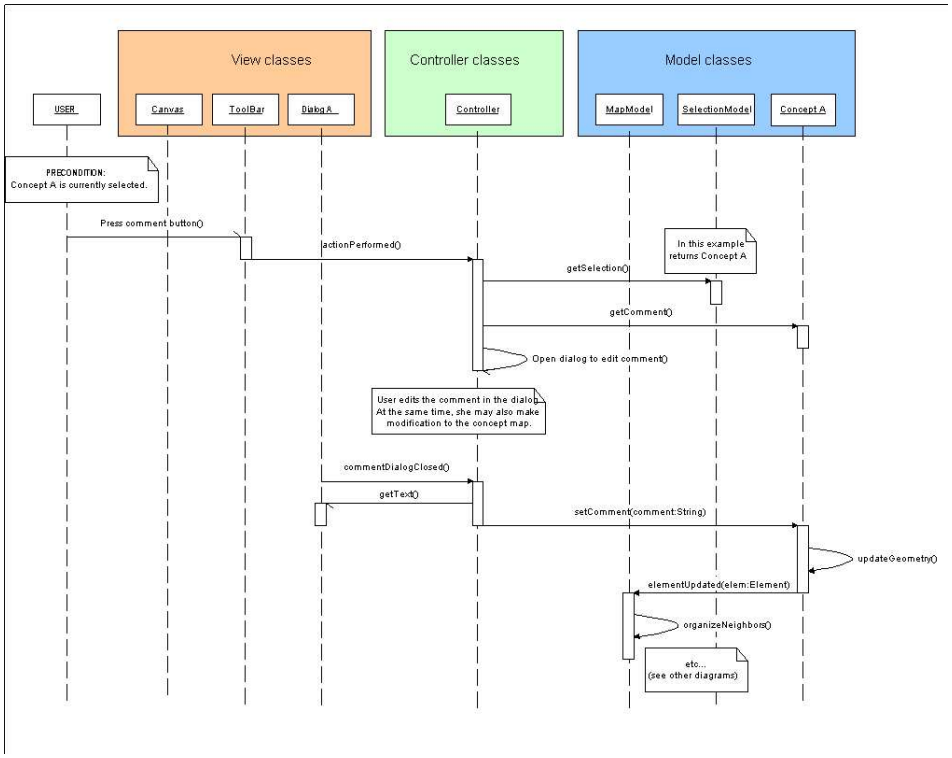
Syötteiden selitykset

syöte	Näppäimistöltä naputeltu kirjoitus
Enter	Enterin painallus
1-k	1 klikkaus kanvaksen päällä
1-e	1 klikkaus elementin päällä
2-k	2 klikkausta kanvaksen päällä
k-raahaus	Käsitteen luonti raahauksella eli raahaus käsitteen yhteyskuvakkeesta kanvakselle
y-raahaus	Yhteyden luonti raahauksella eli raahaus elementin yhteyskuvakkeesta elementille
Del	Deleten painaminen
siirto	elementin siirtäminen
ulkoasu	elementin ulkoasun muokkaaminen
Ctrl + 1-e	Control pohjassa ja hiiren napssauttaminen elementin päällä
alue	alueen valinta eli raahaus

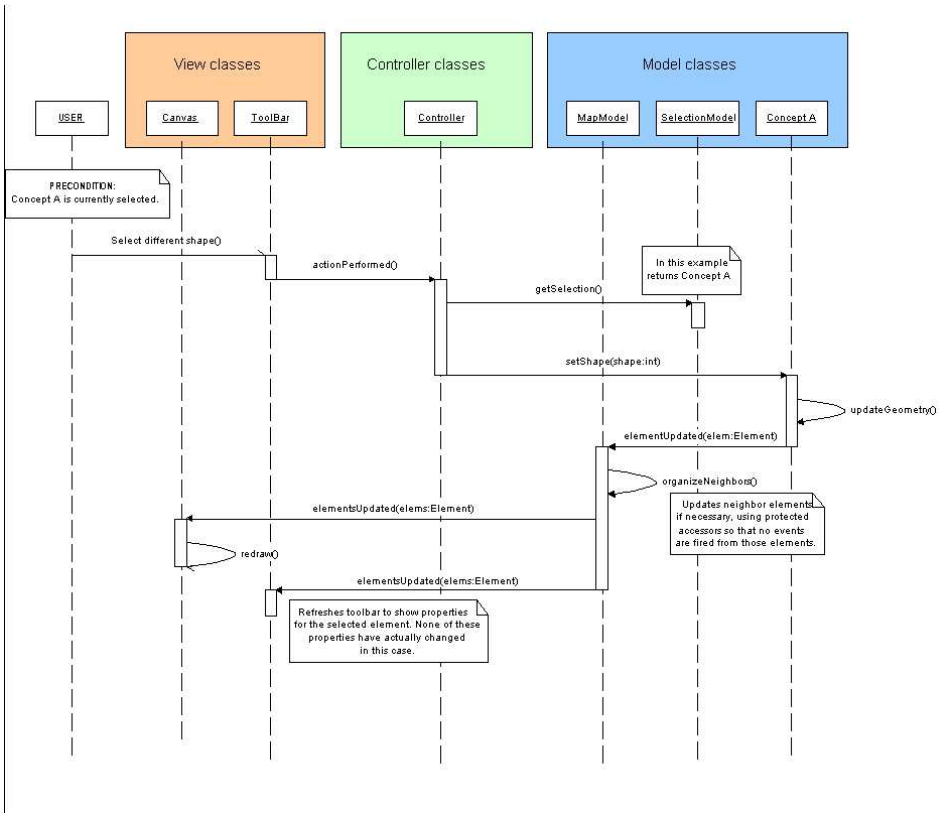
Kuva 5: CoMa-järjestelmän tilasiirtymämatriisi



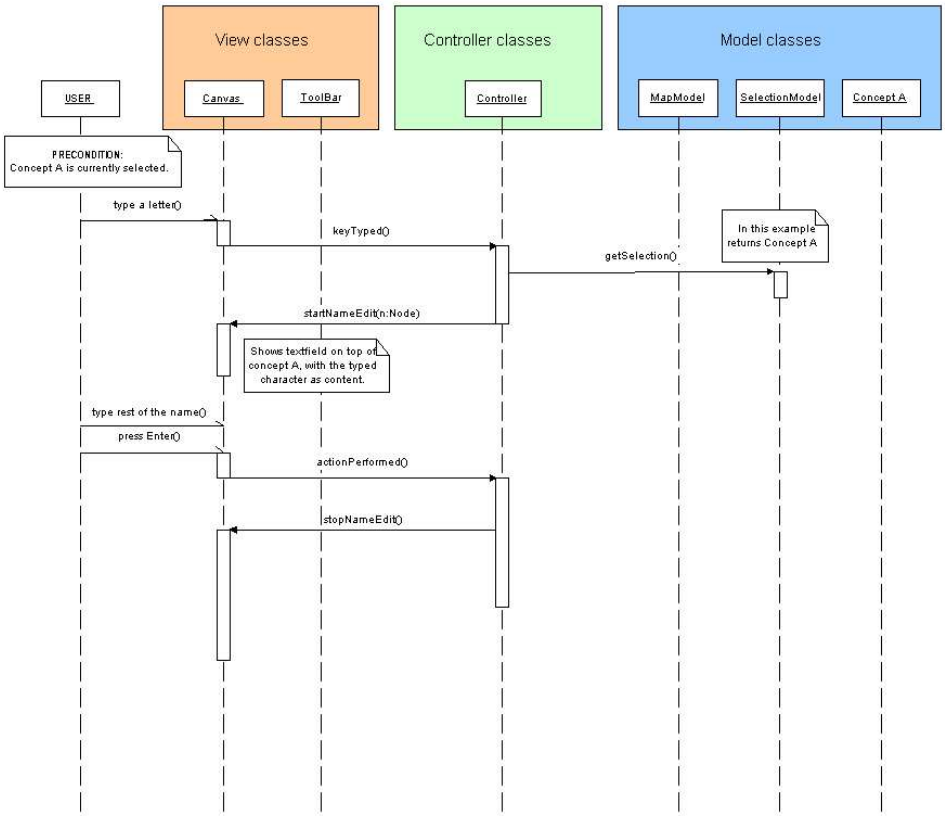
Kuva 6: CoMa-järjestelmän tilasiirtymäkaavio



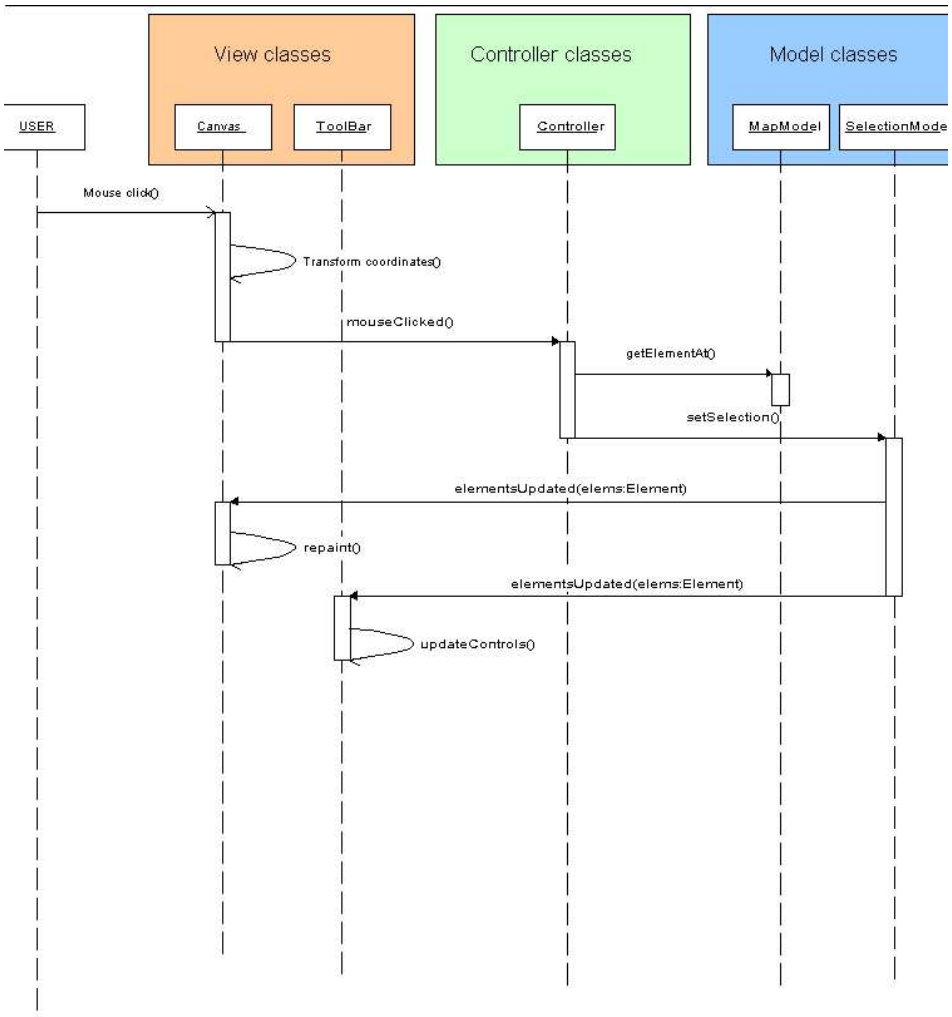
Kuva 7: Kommentin luominen



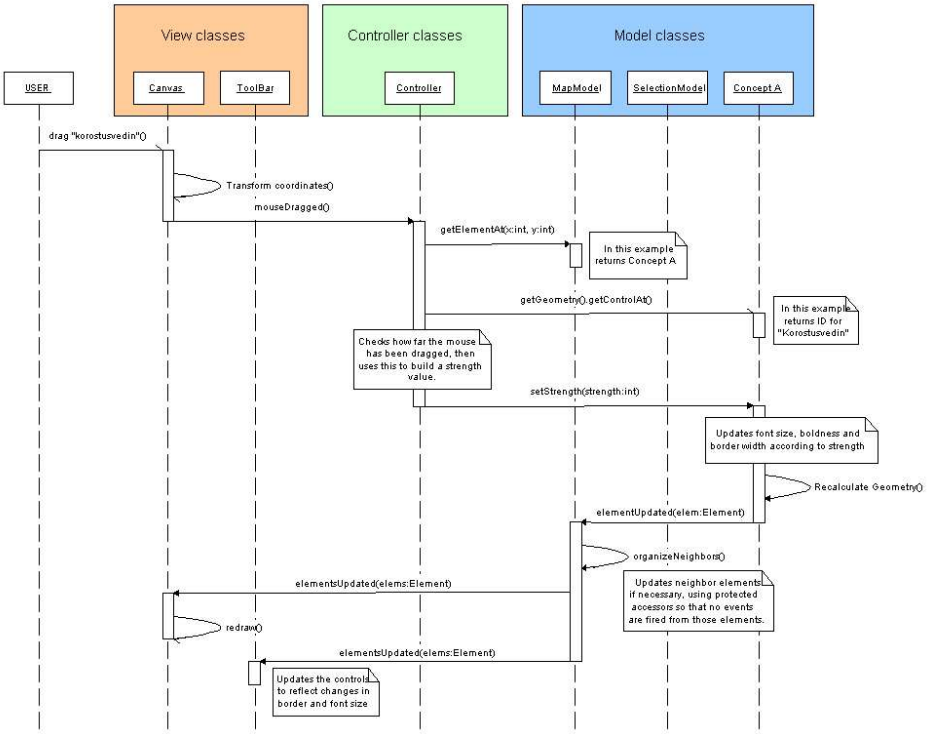
Kuva 8: Käsitteen muodon muuttaminen



Kuva 9: Elementin nimeäminen



Kuva 10: Elementtien valitseminen



Kuva 11: Käsitteen korostaminen korostusraahaimella