

Aleksi Riihiaho

CAD-ohjelmaan liitetyn palkkirakenteiden lujuuslaskentamoduulin käyttöliittymän suunnittelu ja toteutus

Diplomityö

|  |  |
| --- | --- |
| Tarkastaja: professori Reijo Kouhia Tarkastaja ja aihe hyväksytty  Teknisten tieteiden tiedekuntaneuvoston kokouksessa x. xkuuta 2016 |  |
|  |  |

TIIVISTELMÄ

**ALEKSI RIIHIAHO**: CAD-ohjelmaan liitetyn palkkirakenteiden lujuuslaskentamoduulin käyttöliittymän suunnittelu ja toteutus

Tampereen teknillinen yliopisto

Diplomityö, XX sivua, YY liitesivua

Xkuu 2016

Konetekniikan diplomi-insinöörin tutkinto-ohjelma

Pääaine: Koneiden ja rakenteiden analysointi

Tarkastaja: professori Reijo Kouhia

Avainsanat: FEM

Tiivistelmä on suppea

ABSTRACT

**ALEKSI RIIHIAHO**: Heading

Tampere University of Technology

Master of Science Thesis, XX pages, YY Appendix pages

X 2016

Master’s Degree Programme in Mechanical Engineering

Major: Analysis of Machines and Structures

Examiner: Professor Reijo Kouhia

Keywords: FEM

The basics in English in Master’s theses.

ALKUSANAT

Tämä dokumenttipohja on laadittu TTY:n opinnäytetyöohjeen vuoden 2014 version mukaan edellistä pohjaa muokkaamalla. Työryhmä haluaa kiittää kaikkia ohjeen päivitykseen osallistuneita.

Alkusanoissa esitetään opinnäytetyön tekemiseen liittyvät yleiset tiedot. Tapana on myös esittää kiitokset työn tekemiseen vaikuttaneille henkilöille ja yhteisöille. Alkusanat eivät kuulu arvioinnin piriin, mutta niissä ei silti ole sopivaa moittia tai kritisoida ketään. Alkusanojen pituus on enintään 1 sivu. Alkusanojen lopussa on päivämäärä, jonka jälkeen työhön ei ole enää tehty korjauksia.

Tampereella, x.x.2016

Aleksi Riihiaho

SISÄLLYSLUETTELO

[1. Johdanto 1](#_Toc443372246)

[2. Palkkirakenteiden elementtimenetelmä 2](#_Toc443372247)

[2.1 Tekninen taivutusteoria 2](#_Toc443372248)

[2.2 Avaruuspalkkielementti 3](#_Toc443372249)

[2.3 Laskentamalli 3](#_Toc443372250)

[2.4 Aktiivisarakeratkaisija 5](#_Toc443372251)

[3. Käyttöliittymäsuunnittelun periaatteita 6](#_Toc443372252)

[3.1 Käyttäjäkokemus 6](#_Toc443372253)

[3.2 Käyttäjätarpeiden kartoitus 7](#_Toc443372254)

[3.3 Käyttäjäkeskeinen suunnittelu 7](#_Toc443372255)

[3.3.1 Normanin suunnitteluperiaatteet 7](#_Toc443372256)

[3.3.2 Nielsenin heuristiikat 9](#_Toc443372257)

[3.3.3 Shneidermanin kahdeksan kultaista sääntöä 9](#_Toc443372258)

[4. Graafisen käyttöliittymän ohjelmointi Windows-ympäristössä 10](#_Toc443372259)

[4.1 Suunnittelumallit 10](#_Toc443372260)

[4.2 MVC-arkkitehtuuri 10](#_Toc443372261)

[4.3 Windows Ribbon Framework 10](#_Toc443372262)

[4.4 OpenGL-piirtotyökalut 10](#_Toc443372263)

[5. Vaatimukset toteutettavalle käyttöliittymälle 11](#_Toc443372264)

[5.1 Asiakasvaatimukset 11](#_Toc443372265)

[5.2 STAFRA 12](#_Toc443372266)

[5.3 Yhteensopivuus Vertexin eri ohjelmistojen kanssa 13](#_Toc443372267)

[6. Käyttöliittymän toteutus 14](#_Toc443372268)

[6.1 Tietorakenne 14](#_Toc443372269)

[6.2 Puunäkymä 14](#_Toc443372270)

[6.3 Ribbon-valikko 14](#_Toc443372271)

[6.4 Käyttötapaukset 14](#_Toc443372272)

[6.4.1 Tutkimusten luominen ja poistaminen 14](#_Toc443372273)

[6.4.2 Tuentojen ja kuormitusten asettaminen 14](#_Toc443372274)

[6.4.3 Tulosten tarkastelu 14](#_Toc443372275)

[7. Jatkokehitystarpeet 15](#_Toc443372276)

[8. Yhteenveto 16](#_Toc443372277)

[Lähteet 17](#_Toc443372278)

LIITE A:

LYHENTEET JA MERKINNÄT

CC-lisenssi Creative Commons -lisenssi

LaTeX ladontajärjestelmä tieteelliseen kirjoittamiseen

SI-järjestelmä ransk. Système international d’unités, kansainvälinen mittayksikköjärjestelmä

TTY Tampereen teknillinen yliopisto

URL engl. Uniform Resource Locator, verkkosivun osoite

*a* kiihtyvyys

**F**voima

*m* massa

Työssä käytetyt lyhenteet ja merkinnät määritellään ja selitetään kootusti aakkosjärjestyksessä työn alussa ja kun ne esiintyvät tekstissä ensimmäisen kerran Lyhenteiden kanssa käytetään tällöin sulkeita. Selitetekstin tyyli on tässä *Symbol description*. Tämän sivun lopussa on *Section Break*, jotta sivunumerointi menee oikein. Lisäksi johdannon yläotsakkeen (header) asetus *Link to Previous* on pois päältä, ja lisäksi sivunumeron muotoilusta on valittu *Start at 1* (eikä *Continue*).

.

# Johdanto

Vertex

Vertex-ohjelmistot

Miksi tehdään sisäänrakennettu FEM-laskentaosio Vertexin ohjelmistoihin?

Käyttöliittymän suunnittelu ja toteutus FEM-laskentaosiolle

Minkälaisia ongelmia on tarkoitus laskea? Asiakastarve

Vertexin aiemmat FEM-laskentamoduulit, Numerola

# Palkkirakenteiden elementtimenetelmä

Elementtimenetelmä (*Finite element method*, FEM) on matemaattinen menetelmä differentiaali- ja osittaisdifferentiaaliyhtälöiden reuna-arvotehtävien numeeriseen ratkaisemiseen. Elementtimenetelmällä voidaan muuntaa osittaisdifferentiaaliyhtälö lineaariseksi yhtälösysteemiksi, joka on helposti ratkaistavissa. Muunnos tapahtuu diskretisoimalla kenttäfunktion äärettömän suuri tuntematon arvojoukko äärelliseksi määräksi solmuarvoja, jotka voidaan ratkaista elementtimenetelmällä. Eri tekniikan aloilla tämänlaisia tehtäviä joudutaan ratkaisemaan paljon, joten elementtimenetelmä onkin nykyään levinnyt muun muassa mekaniikan, termodynamiikan, virtausmekaniikan, murtumismekaniikan, akustiikan ja sähkötekniikan aloille. [6][11]

PALKKIRAKENTEIDEN (Direct stiffness method)

Palkkirakenteiden elementtimenetelmä eroaa tästä yllä mainitusta yleistetystä elementtimenetelmästä siten, että siinä diskretoitavat osat, eli palkit, voidaan käsittää yksiulotteisiksi. Tämän johdosta palkkirakenteiden elementtimenetelmä voidaan perustaa suoraan rakenteiden mekaniikan analyyttisten ratkaisumenetelmien teorialle. Tämän tarkoittaa myös sitä, että palkkirakenteiden elementtimenetelmä johtaa teknisen taivutusteorian puitteissa tarkkaan ratkaisuun, toisin kuin likiarvoratkaisun tuottava numeerinen elementtimenetelmä. [11]

Tässä luvussa käsitellään teoriaa, jonka tietämystä tarvitaan palkkirakenteiden elementtimenetelmän käyttämisessä.

Alun perin elementtimenetelmä kehittyi lentokone- ja avaruusteollisuuden lujuusanalyysejä tehneiden insinöörien keskuudessa 1950-luvulla, josta se levisi nopeasti muun muassa erinäisiin rakennus- ja konetekniikan sovellutuksiin.

## Tekninen taivutusteoria

Tekninen taivutusteoria, Bathe s.234, Structural Analysis with the Finite Element s.33 [9]

Laskentamallin luominen, tärkeys, mallin tehokkuus, yksinkertaistaminen, Hibbler Idealization [4]

Bathe, laskennan vaiheet

## Avaruuspalkkielementti

Jäykkyysmatriisin johto tasopalkkielementistä (derivointi 3-asteen polynomista, koska 4 vapausastetta)

Leikkausmuodonmuutokset

Timoshenko-palkki, leikkauslukkiutumisen esto

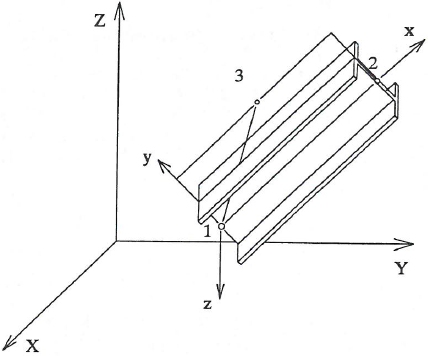
## Laskentamalli

Rakenteiden mekaniikassa elementtimenetelmän perusajatuksena on jatkuvan materiaalikontinuumin diskretisoiminen äärelliseen määrään elementtejä, jotka liittyvät toisiinsa niin sanotuissa solmukohdissa. Rakenteen muodostamaa elementtijoukkoa kutsutaan laskentamalliksi.

Elementtien väliset solmut voidaan luokitella kahteen kategoriaan: lokaalisolmuihin ja globaalisolmuihin. Lokaalisolmuilla tarkoitetaan yksittäisen elementin päissä tai nurkissa olevia solmuja, joilla elementit liittyvät toisiin elementteihin. Globaalisolmuilla taas tarkoitetaan koko laskentamallin solmuja. Globaalisolmut yksilöidään yleensä juoksevalla numeroinnilla, mutta numeroinnin järjestyksellä ei ole laskennan kannalta väliä. Elementin suunnistus määrää sen alku- ja loppupään. Yksittäisen elementin lokaalisolmut numeroidaan järjestyksessä pienimmästä suurimpaan alkupäästä lähtien.

Lokaalikoordinaatistolla tarkoitetaan yksittäisen elementin paikallista koordinaatistoa, jossa x-akseli osoittaa palkin suuntaan alkupäästä loppupäähän päin. Y- ja Z-akselit valitaan yleensä palkin poikkileikkauksen pääsuuntien mukaan. Koko rakenteelle yhteiselle koordinaatistolle käytetään nimitystä globaalikoordinaatisto. Solmumittausjärjestelmällä tarkoitetaan kussakin tilanteessa voimassa olevaa koordinaatistoa, jossa solmumittaus suoritetaan.

Avaruuspalkkielementti on 2-solmuinen palkkielementti, jonka molemmilla solmuilla on 6 vapausastetta. Kuva 1 on esitetty avaruuspalkkielementti, sen lokaalikoordinaatisto ja globaalikoordinaatisto. Elementin suunnistus kulkee solmusta 1 solmuun 2. Palkin yläpuolella oleva solmu 3 toimii suuntasolmuna, joka määrittää elementin lokaalikoordinaatiston y-akselin suunnan.



1. Avaruuspalkkielementti ja sen lokaalikoordinaatisto [13]

”Perusyhtälö”

Jäykkyysmatriisi

Sijoittelusummaus

Palkkielementtien solmumittausjärjestelmää joudutaan usein kiertämään, koska palkit voivat osoittaa globaalikoordinaatistossa mielivaltaiseen suuntaan. Jotta sijoittelusummaus voidaan suorittaa, tulee yksittäisten elementtien ja koko rakenteen solmumittausten olla samansuuntaiset. Koordinaatiston kierto elementin lokaalikoordinaatistosta globaalikoordinaatistoon tehdään niin sanotulla koordinaatiston kiertomatriisilla. Kiertomatriisi kootaan siten, että lokaalikoordinaatiston kantavektoreilla **i**, **j** ja **k** lausutun vektorin **v** sekä globaalikoordinaatiston kantavektoreilla **i**, **j** ja **k** lausutun vektorin **v** välillä on lineaarinen yhteys:

. (1)

KAAVA kiertomatriisista ja termeistä

Solmusuureet, siirtymien ja voimien mittaus

Solmut

Sauvaelementti

Palkkielementti

Laskentamalli

Pistekuormitukset

Kenttäkuormitukset

## Aktiivisarakeratkaisija

Stafra

# Käyttöliittymäsuunnittelun periaatteita

Asd

## Käyttäjäkokemus

Käyttäjäkokemuksella tarkoitetaan käyttäjän subjektiivista vaikutelmaa tuotteesta, joka syntyy käyttäjän ollessa vuorovaikutuksessa sen kanssa. Käyttäjäkokemus on siis tunnereaktioita, uskomuksia ja mieltymyksiä, jotka ilmenevät ennen tuotteen käyttöä, käytön aikana ja käytön jälkeen. Täsmällinen määritelmä käyttäjäkokemukselle löytyy ISO 9241-210 -standardista: ”Henkilön havainnot ja vasteet, jotka ovat seurausta tuotteen, järjestelmän tai palvelun käytöstä ja/tai ennakoidusta käytöstä”. [14]

Käyttäjäkokemukseen voidaan vaikuttaa luonnollisesti tuotteen käytettävyyttä parantamalla. Käytettävyyden kannalta oleellisinta on usein käyttöliittymän hyvä ja huolellinen suunnittelu. Myös käytettävyydelle on luotu standardin mukainen määritelmä (ISO 9241-11) ja se kuuluu näin: ”Tarkoituksenmukaisuus, tehokkuus ja tyytyväisyys, jolla tuotteen määritellyt käyttäjät saavuttava määritellyt tavoitteet tietyissä käyttöympäristöissä”. Käytettävyyden lisäksi käyttäjäkokemukseen vaikuttavat myös tuotteen esteettiset ominaisuudet kuten esimerkiksi visuaalisuus ja auditiivisuus. Tuotteeseen liittyvillä oheistarvikkeilla- ja palveluilla on myös vaikutuksensa käyttäjäkokemuksen syntymisessä. Tällaisia voivat olla esimerkiksi tuotteen käyttöohje tai tukipalvelu. [14]

Tuotteen käyttöympäristö vaikuttaa merkittävästi käyttäjäkokemuksen syntymisessä. Käyttöympäristö, eli käyttökonteksti, määrittelee minkälaisessa ympäristössä ja olosuhteissa tuotetta käytetään. Sen voidaan katsoa koostuvan tehtäväkohtaisesta, fyysisestä, sosiaalisesta ja teknisestä ympäristöstä. Teknisestä ympäristöstä esimerkkinä käy vaikkapa laite, jolla kehitettävää ohjelmistoa on suunniteltu käytettävän. Ohjelmiston käyttöliittymää suunniteltaessa on tiedettävä tarkkaan, minkälaisilla laitteilla loppukäyttäjät tulevat tuotetta käyttämään. Sosiaalisia käyttöympäristöjä ovat esimerkiksi koti ja työpaikka. Kotona tunnelma on usein rentoutunut kun taas työpaikalla ilmapiiri voi olla virallisempi. [14]

Käytettävyydeltään hyvän käyttöliittymän ominaisuuksia



1. Käyttäjäkokemuksen syntyminen käyttäjän ja tuotteen välisessä vuorovaikutuksessa [14]

Miksi vaalia käyttäjäkokemusta? Ohjelmistonäkökanta

## Käyttäjätarpeiden kartoitus

## Käyttäjäkeskeinen suunnittelu

Prosessi, jolla hyvä (ISO 9241-11) tuote suunnitellaan

Näkökulma, joka pyrkii ymmärtämään ihmisen käyttäytymistä sekä taitoja ja hyödyntää niitä tuotesuunnittelussa.

Asiakaslähtöisyys

Käyttäjäkokemuksen suunnittelu on tuotteen käyttäjälle ilmeneviä laatuominaisuuksia

Paperikäyttöliittymällä prototypointi

Seuraavissa aliluvuissa 3.3.1 – 3.3.3 käsitellään erilaisia hyvän käyttäjäkeskeisen suunnittelun periaatteita, jotka ovat sovellettavissa käyttöliittymäsuunnitteluun. Suunnitteluperiaatteet soveltuvat myös käytettävyyden heuristiseen arviointiin. [14]

### Normanin suunnitteluperiaatteet

Käyttäjäkeskeisen suunnittelun alalla uraa tehnyt professori Donald A. Norman kertoo kirjassaan ”The Design of Everyday Things” (2013) [8] kuinka jokapäiväisten esineiden käytettävyyttä voitaisiin parantaa. Näitä ohjeistuksia on helppo soveltaa myös ohjelmistojen käyttöliittymien suunnitteluun. Norman on koonnut ohjeensa viideksi suunnitteluperiaatteeksi:

1. Loogiset kytkennät

Käyttäjälle tulisi selvitä, mikä ohjaimen ja toiminnan välinen kytkentä on. Kytkentä tulisi olla selvä jo ennen kuin käyttäjä on ensimmäistä kertaa käyttänyt ohjainta. Loogisia kytkentöjä suunniteltaessa kannattaa käyttää hyväksi fyysisiä ja tilaan liittyviä analogioita. Esimerkiksi ylöspäin suuntautuva liike ja ylöspäin osoittava nuoli merkitsevät usein lisäystä kun taas alaspäin suuntautuva liike ja alaspäin osoittava nuoli yhdistetään usein vähentymiseen. Tällaista logiikkaa voidaan käyttää hyväksi esimerkiksi lukumäärää tai mittasuhteita kuvaavissa ohjaimissa.

1. Mahdollisuudet ja rajoitteet

Ihmisen muodostaessa käsitystä tuotteen toiminnasta se tekee näkemästään rakenteesta tulkintoja sen mahdollisuuksien ja rajoitteiden perusteella. Mahdollisuudet tarkoittavat eri toimintavaihtoehtoja, joita ovat esimerkiksi käyttöliittymän nappulat, jotka ovat painettavissa. Rajoitteilla puolestaan tarkoitetaan toiminnalle asetettuja rajoituksia, joita ovat esimerkiksi tekstikenttään syötettävän merkkijonon maksimipituus tai harmaannutetut, ei-käytössä olevat käyttöliittymän nappulat.

1. Näkyvyys

Näkyvyydellä tarkoitetaan käyttäjän kannalta oleellisten tietojen ja toimintojen esille tuomista. Käyttöliittymän tulee olla visuaaliselta ilmeeltään sellainen, että käyttäjän on helppo ymmärtää, kuinka tuotetta on tarkoitus käyttää toivotun lopputuloksen saavuttamiseksi. Oikeiden elementtien on oltava käyttöliittymässä näkyviä, jotta käyttäjä saa oikean käsityksen tuotteen toimintatavasta.

1. Palaute

Palaute on käyttäjälle annettavaa tietoa tuotteen toiminnasta. Palautteesta ilmenee, mitä on tehty ja mitä tuloksia saavutettu. Palaute on oleellista onnistuneen toiminnan jälkeen, mutta erityisesti myös virhetilanteissa palaute on tärkeää. Olennaista palautteessa on vasteaika. Palautteen on tapahduttava heti käyttäjän toiminnan jälkeen, jotta se osataan yhdistää kyseiseen toimintaan.

1. Virheiden käsittely

Käyttöliittymäsuunnittelun tavoitteena on toki tehdä käyttäjän tekemät virheet mahdottomiksi, mutta se harvoin onnistuu käytännössä. Tämän takia virhetilanteiden käsittely ja niistä palautuminen on keskeistä käyttöliittymäsuunnittelussa. Käyttäjän ei tulisi koskaan syyttää itseään virheistä, sillä se voi johtaa turhautumiseen ja jopa tuotteen hylkäämiseenkin. Käyttäjälle tulee tarjota virhetilanteen sattuessa selkeät ohjeet, miten tilanteesta pääsee pois.

### Nielsenin heuristiikat

Käytettävyysasiantuntija Jakob Nielsen esittelee kirjassaan ”Usability Engineering” (1993) [7] kymmenen käytettävyysheuristiikan listan, jonka avulla voi arvioida tuotteen tai käyttöliittymän käytettävyyttä. Lista soveltuu myös uuden tuotteen tai käyttöjärjestelmän suunnittelun periaatteiksi. Nielsenin heuristisen arvioinnin listaan kuuluvat:

1. Järjestelmän tilan näkyvyys
2. Järjestelmän ja todellisen maailman yhteys
3. Käyttäjän kontrolli ja vapaus
4. Johdonmukaisuus
5. Virheiden välttäminen
6. Tunnistettavuus ennen muistamista
7. Käytön joustavuus ja tehokkuus
8. Yksinkertaisuus
9. Virheiden käsittely
10. Opasteet ja dokumentaatio

### Shneidermanin kahdeksan kultaista sääntöä

Tietojenkäsittelytieteiden professori Ben Shneidermanin kirjassa ”Designing the user interface: strategies for effective human-computer interaction” (1998) [12] esitellään kahdeksan kultaista sääntöä erityisesti ohjelmistojen käyttöliittymien dialogien suunnitteluun ja arviointiin. Säännöt ovat kuitenkin sovellettavissa ohjenuoriksi myös koko järjestelmän suunnitteluun [14]. Shneidermanin kahdeksan kultaista sääntöä ovat:

1. Noudata yhtenäisyyttä toimintaketjuissa ja toimintatavoissa
2. Tarjoa edistyneille käyttäjille oikoteitä
3. Tarjoa informatiivista palautetta
4. Suunnittele dialogit siten, että ne johtavat lopputulokseen
5. Tarjoa yksinkertaista virheenkäsittelyä
6. Auta käyttäjää toipumaan virhetilanteista
7. Tue käyttäjän kontrollin tunnetta
8. Rajoita käyttäjän lyhytkestoisen muistin kuormitusta

# Graafisen käyttöliittymän ohjelmointi Windows-ympäristössä

Tässä luvussa käsitellään ohjelmistotekniikkaan liittyviä käsitteitä ja malleja, jotka koskevat erityisesti graafisen käyttöliittymän ohjelmointia Windows-ympäristössä C++-ohjelmointikielellä. Luvussa painotetaan erityisesti niitä asioita, joiden ymmärtäminen on tässä diplomityössä suunniteltavan ja toteutettavan graafisen käyttöliittymän kannalta oleellisia.

METATEKSTIÄ asian järjestyksestä

## Olio-ohjelmointi

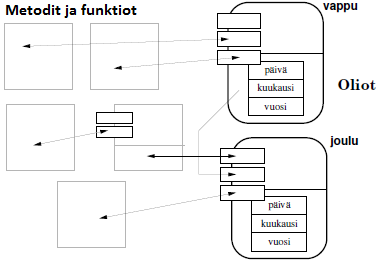
Ohjelmistotekniikan yhtenä vaikeimpana ongelmana pidetään suurten ohjelmistojen tekemisen monimutkaisuutta. Tietokonelaitteistojen tekniikoiden räjähdysmäinen kehitys on mahdollistanut entistä suurempien ja kompleksisempien ohjelmistojen suorittamisen. Ohjelmistojen tekemiseen tarvittavat työkalut ja menetelmät, kuten ohjelmointikielet, eivät ole kuitenkaan kehittyneet samaa vauhtia laitteistotekniikan kehityksen mukana. Tämän seurauksena on puhjennut niin sanottu ohjelmistokriisi, jolla tarkoitetaan juuri tätä tietokonelaitteistojen ja ohjelmointityökalujen kehityksien epävakaata suhdetta. [10]

Yhtenä ratkaisuna ohjelmistokriisin ratkaisemiseksi on tarjottu oliokeskeisiä menetelmiä, jotka ovatkin viime aikoina olleet suurimman huomion kohteena ohjelmistoteollisuudessa. Oliokeskeisiä menetelmiä käyttävässä ohjelmoinnissa, eli lyhyemmin olio-ohjelmoinnissa, perusperiaatteena on ongelmien jakaminen yhden ihmisen hallittaviin osakokonaisuuksiin sekä yksinkertaistaminen abstrahoimalla. Abstrahoinnin voidaan sanoa tarkoittavan toisiinsa liittyvien asioiden keräämistä yhdeksi järkeväksi toiminnalliseksi kokonaisuudeksi, joka voi olla puhtaasti ajatuksellinen eli abstrakti. [10]

### Oliot ja luokat

Olio-ohjelmoinnissa ongelmien jako yhden ihmisen hallittaviin osakokonaisuuksiin tarkoittaa ohjelmakoodin toiminnallisesti sekä tietorakenteellisesti yhteenkuuluvien osien kokoamista olioiksi. Yhden olion rakenne koostuu tietorakenteesta, sitä käsittelevästä ohjelmakoodista sekä olion tietorakenteen käsittelyyn tarkoitetusta julkisesta rajapinnasta. Oliorakenteen toteuttajan vastuulla on olion sisäinen toteutus, eli kuinka sen tietorakenne koostetaan ja kuinka sitä käsitellään. Oliorakenteen toteuttaja suunnittelee oliolle myös julkisen rajapinnan. Julkisen rajapinta tulisi olla mahdollisimman selkeä ja ymmärrettävä, vaikka sen käyttäjä ei tuntisi olion sisäistä toteutusta lainkaan. Julkinen rajapinta tarkoittaa metodeja tai funktioita, joilla oliota tai sen tilaa voidaan käsitellä sen ulkopuolelta muualla ohjelmakoodissa. Olio siis kätkee sisäänsä tietorakenteensa ja sen käsittelyn, jolloin olion käyttäjän vastuulle jää ainoastaan olion julkisen rajapinnan oikeaoppinen käyttö. Olion käytön kannalta epäoleellisen tiedon kapseloiminen sen sisälle on yksi tärkeimpiä abstrahoinnin työkaluja ohjelmoinnissa. [10]

C++-kielessä oliot luodaan luokkien avulla. Luokka on määrittely siitä muodostettavan olion tietorakenteelle, metodeille ja funktioille sekä julkiselle rajapinnalle. Luokka on siis yleinen rakenne, joka edustaa kaikkia kyseisestä luokasta luotuja olioita. Ohjelman ajon aikana ei ole olemassa luokkia, vaan ainoastaan luokan kuvauksen perusteella luotuja olioita. [10]



1. Oliot ja niiden tietorakenne sekä julkinen rajapinta. Kuva muokattu lähteestä [10].

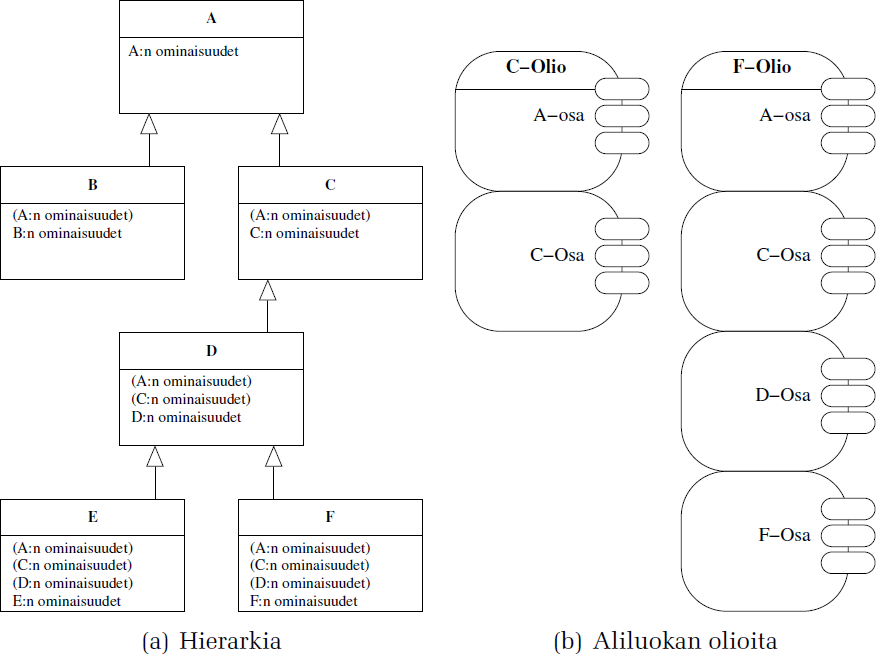
Kuvassa 3 on havainnollistettu tyypillisiä olio-ohjelmoinnissa käytettäviä olioita. Oliot ”vappu” ja ”joulu” ovat samasta luokasta tehtyjä instansseja, ja niiden tarkoitus on esittää päiväystä. Päiväystä esittävän olion tietorakenteeseen kuuluu siis tässä tapauksessa päivä, kuukausi ja vuosi, jotka voidaan ilmaista kokonaislukuina. Olioiden julkisessa rajapinnassa on metodeita ja funktioita, joiden avulla niiden tilaa pystytään muokkaamaan ja kontrolloimaan olioiden ulkopuolelta. Kuvassa olevien päiväystä esittävien olioiden tapauksessa metodit voisivat olla esimerkiksi päiväyksen kasvattaminen halutulla määrällä päiviä tai päiväyksen tulostaminen näytölle tietyllä syntaksilla.

Ohjelmistoja suunniteltaessa olio-ohjelmointi on oiva työkalu ohjelmointiongelman mallintamiseen siten, että se vastaisi mahdollisimman hyvin todellisen maailman ongelmaa. Kun ohjelmassa olevat oliot vastaavat mahdollisimman hyvin todellisen maailman olioita, on monimutkaisenkin tietokoneohjelman rakenne helpommin ymmärrettävissä. Toisaalta ohjelmistoihin suunniteltavien olioiden tulisi olla myös ohjelmoinnin kannalta katsottuna käytännöllisiä ja tehokkaita, jolloin ne eivät välttämättä täsmällisesti vastaa todellisen maailman olioita. Olio-ohjelmointi vaatii ohjelmoijalta hyvää abstraktista ajattelukykyä. [10]

### Periytyminen

Yksi tietokoneohjelmien rakenteiden hahmottamisen ja kategorisoimisen avuksi luotu mekanismi on oliokeskeisissä ohjelmointikielissä yleisesti mukana oleva periytyminen. Se tarkoittaa luokkien jaottelua kantaluokkiin ja aliluokkiin niiden yhteisten ominaisuuksien ja sukulaisuussuhteiden perusteella. Periytymisen avulla ohjelmiston olioille on mahdollista luoda hierarkkinen periytymisrakenne, joita esiintyy usein myös todellisessa maailmassa. Periyttämisen avulla voidaan luoda useita samankaltaisia olioita, joilla on hieman toisistaan poikkeava käyttäytyminen, kirjoittamatta kuitenkaan uudestaan olioiden yhtenevän käyttäytymisen toteuttavaa koodia. [10]

Käytännössä C++-ohjelmointikielessä periytymisellä tarkoitetaan sitä, että uusi luokka muodostetaan toisen olemassa olevan luokan pohjalta siten, että uusi luokka sisältää kaikki toisen luokan ominaisuudet. Tällöin alkuperäistä luokkaa, jonka ominaisuudet periytyvät uuteen luokkaan, kutsutaan kantaluokaksi. Uutta, periytettyä luokkaa, kutsutaan aliluokaksi.



1. Periytymishierarkia ja oliot [6].

Kuvassa 4 on esimerkki luokkien periytymishierarkiasta havainnollistavana kaaviona. Vasemman puolen hierarkiassa nuoli osoittaa aina aliluokasta kantaluokkaan päin. Esimerkin tapauksessa luokka A toimii kantaluokkana kaikille muille luokille. Luokkien ominaisuudet, eli julkisen rajapinnan tarjoamat metodit ja funktiot, periytyvät kuvan vasemman puolen kaavion mukaisesti. Aliluokista luotujen olioiden voidaan ajatella sisältävän kaikki sen periytymishierarkiassa yläpuolella olevien sukulaisluokkien informaation, kuten kuvan oikealla olevassa kuviossa on esitetty.

## Suunnittelumallit

### MVC-arkkitehtuuri

### Visitor pattern -malli

## Windows Ribbon Framework

## OpenGL-piirtotyökalut

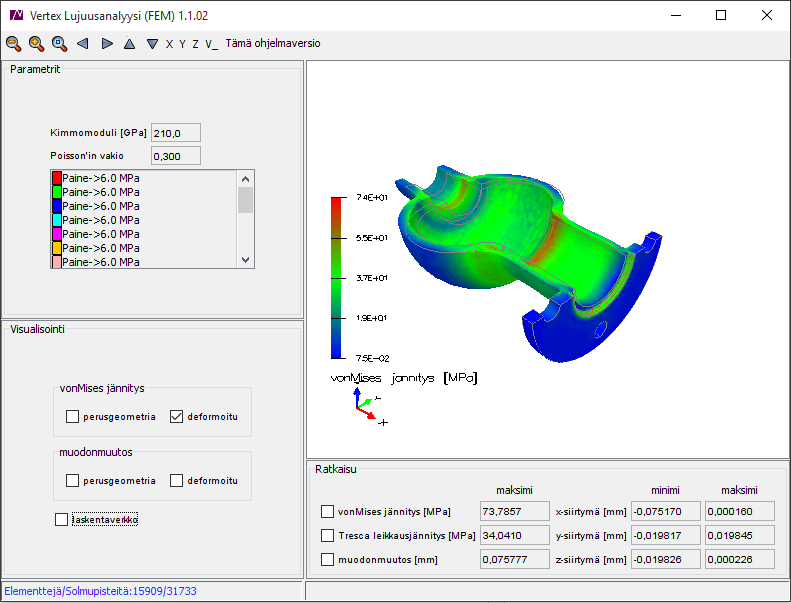
# Vaatimukset laskentamoduulille

Vertex-ohjelmistoihin on aiemminkin sisällytetty lujuuslaskentamoduuleja, joihin tässä diplomityössä käsiteltävä palkkirakenteiden lujuuslaskentamoduuli on jatkoa. Vertex BD -rakennussuunnitteluohjelmistossa on ollut käytössä mistä lähtien kattoristikoiden normin mukainen mitoituslaskenta. mille normeille. Lisäksi Vertex G4 -mekaniikkasuunnitteluohjelmassa on ollut käytössä Numerola Oy:n kanssa yhteistyössä toteutettu lujuuslaskentamoduuli yksittäisille tilavuusmalleille.

KUVA Vertex BD Truss engineering

Voimat asetetaan z-suunnassa. Voimat asetetaan piirustukseen mielivaltaiseen kohtaan. BD:n kuva-mallipari.

Ongelma: muodostettua FEM-mallia (STAFRA:an menevää) ei nähdä missään muodossa, eikä sitä voi muokata



1. Vertex G4:n yksittäisen tilavuusmallin lujuusanalyysi.

KUVA Vertex G4 Numerola

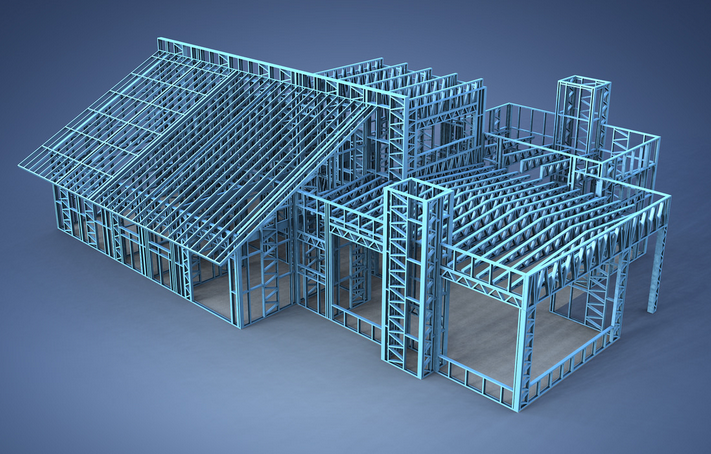
Tällä hetkellä Numerola: ei verkota oikein, kankea käyttöliittymä, ei ole yhtenäinen osa Vertexiä.

3D mesh, Spatial

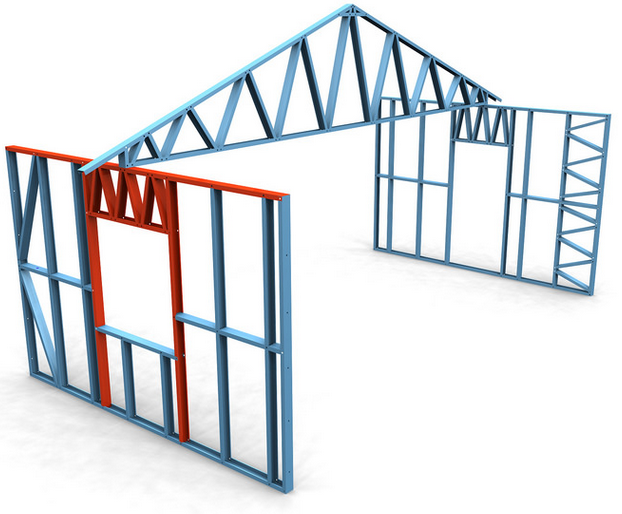
Tällä hetkellä STAFRA BD: ainoastaan ristikkolaskentaan räätälöity, ei voida laskea esimerkiksi aukkopalkkeja (lintel-rakenteita)

## Asiakasvaatimukset

Vertex yksi suurimmista CFS-profiilirakentamisen suunnittelutyökaluista. Tällä hetkellä voidaan laskea vain ristikoita. Ei mahdollisuuksia laskea esimerkiksi seiniä, aukkopalkkeja (lintel-rakenteet) tai muutakaan. BD: Tarve laskea lintel-rakenteita (palkki pilarien päällä, esim. ovi- tai ikkuna-aukko). Suuret ristikot ja avaruuskehät, joissa todella paljon osia. Teräsrankarakentaminen Australiassa, CFHS-palkit



1. Esimerkki tyypillisestä Vertex BD -ohjelmalla suunnitellusta teräsrankarakenteesta.



1. CFS-profiilirakenteinen ikkunan aukkopalkki.

Vertex G4:ssä nykyään oleva Numerola Oy:n kanssa yhteistyössä tehty lujuuslaskentamoduuli tullaan myöhemmin korvaamaan Vertexin omalla yksittäisiä tilavuusmalleja ja mahdollisesti myös kokoonpanomalleja tukevalla lujuuslaskentamoduulilla. Vertexin asiakaskunnalta kyseltiin, minkälaisia kokemuksia heillä oli nykyisestä lujuuslaskentaoptiosta. Asiakkailta saatiin paljon kehitysehdotuksia ja toiveita tuleviin versioihin. Alla on listattu eniten esiin nousseita toiveita:

* Palkkirakenteiden siirtymien, rasitusten ja tukivoimien laskenta
* Nurjahdusanalyysit
* Putkistojen jännitysanalyysit
* 3D- ja FEM-mallin välinen keskinäinen synkronointi malleja päivitettäessä
* FEM-mallin tekeminen ilman 3D-mallia
* Ohutlevyosien analysoiminen kuorielementeillä
* Kokoonpanomallien analysoiminen solidielementeillä
* Väsymisanalyysi

Seuraaviin Vertexin versioihin tuleva palkkirakenteiden lujuuslaskentamoduuli tulee tyydyttämään asiakkaiden tarpeet palkkirakenteiden siirtymien, rasitusten ja tukivoimien laskennan osalta. Osa asiakkaista oli yrittänyt laskea pienimuotoisia putkipalkkirakenteita tekemällä kokoonpanomallista yhden tilavuusmallin ja syöttämällä sen nykyiseen lujuusanalyysiin. FEM-malli luotiin tällöin solidielementeillä, mikä ei palkkirakenteita analysoitaessa ole hyvä ratkaisu. Palkkielementtejä käyttämällä FEM-malli saadaan luotua huomattavasti pienemmällä määrällä elementtejä ja laskenta on silloin paljon tehokkaampaa.

Muihin asiakaskunnalta saatuihin toiveisiin ei vielä seuraavissa Vertexin versioissa pystytä vastaamaan. Ne tullaan ottamaan kuitenkin huomioon tulevia kehitysaskeleita suunniteltaessa. Yksittäisiä tilavuusmalleja sekä kokoonpanomalleja tukevan solidielementtejä käyttävän lujuuslaskentamoduulin laskentamoottorin prototyyppiä on kehitetty jo varsin pitkälle.

## STAFRA

STAFRA *(Static analysis of frames)* on Lujuustekniikka Oy:n (nykyisin Vertex Systems Oy) kehittämä avaruuskehien statiikan ongelmien ratkaisuun tarkoitettu laskentamoottori. Se perustuu luvussa 2 esiteltyyn palkkirakenteiden elementtimenetelmään ja sen elementit ovat Euler-Bernoulli -palkkiteorian mukaisia palkkeja. Palkkiteoriasta poiketen STAFRA ottaa huomioon myös leikkausjännityksistä aiheutuvat liukumat tehollisten leikkauspinta-alojen avulla. Teholliset leikkauspinta-alat palkin lokaalikoordinaatiston y- ja z-suunnissa saadaan kaavoilla

ja (2)

palkkielementin lokaalikoordinaatiston x-akselin yhtyessä poikkileikkausten pintakeskiöihin sekä y- ja z-akselit poikkileikkausten pääneliöakseleihin. Kertoimet ja aiheutuvat leikkausjännityksen epätasaisesta jakaantumisesta poikkileikkauksen alueelle ja ne ovat riippuvaisia poikkileikkauksen muodosta. STAFRA rajoittuu elementtien materiaalien osalta homogeenisiin, lineaarisesti kimmoisiin ja isotrooppisiin materiaaleihin. [13]

STAFRA on käytössä Vertex BD:n nykyisessä kattoristikoiden mitoituslaskennassa.

## Yhteensopivuus Vertexin eri ohjelmistojen kanssa

Toteutettavan palkkirakenteiden lujuuslaskentamoduulin tulee toimia yhdenmukaisella tavalla sekä Vertex G4 että Vertex BD -ohjelmissa. Tämä aiheuttaa haasteita laskentamoduulin toteutuksessa, sillä G4 ja BD -ohjelmien kehityshistoriat ovat varsin erilaiset. Tämän vuoksi joudutaan tekemään osittain erilaisia toteutuksia samoille laskentamoduulin toiminnallisuuksille.

Vertexin profiilit, KUVA periytymishierarkia

G4, G4 Plant, BD

# Käyttöliittymän toteutus

Asd

## Tietorakenne

## Puunäkymä

## Ribbon-valikko

## Käyttötapaukset

### Tutkimusten luominen ja poistaminen

### Tuentojen ja kuormitusten asettaminen

### Tulosten tarkastelu

# Jatkokehitystarpeet

Asd

# Yhteenveto

Asd

Lähteet

1. K.J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982, 735 p.
2. M. Bordegoni, C. Rizzi, *Innovation in Product Design: From CAD to Virtual Prototyping*, Springer-Verlag London Limited, 2011, 188 p.
3. R.D. Cook, D.S. Malkus, M.E. Plesha, R.J. Witt, *Concepts and Applications of Finite Element Analysis*, John Wiley & Sons, Inc, 1989, 630 p.
4. R.C. Hibbeler, *Structural Analysis*, Pearson Prentice Hall, 2012, 695 p.
5. M. Hirz, W. Dietrich, A. Gfrerrer, J. Lang, *Integrated Computer-Aided Design in Automotive Development*, Springer-Verlag Berlin Heidelberg, 2013, 466 p.
6. K. Koskimies, *Oliokirja*, Talentum Oyj, 2000, 422 s.
7. R. Kouhia, M. Tuomala, *Johdatus mekaniikan ja sähkömagnetiikan numeerisiin menetelmiin*, luentomoniste, Tampereen teknillinen yliopisto, 2014, 417 s.
8. J. Nielsen, *Usability engineering,* Academic Press, 1993, 362 p.
9. D. Norman, *The Design of Everyday Things – Revised and Expanded Edition*, Basic Books, 2013, 347 p.
10. E. Oñate, *Structural Analysis with the Finite Element Method. Linear Statics. Volume 2. Beams, Plates and Shells*, Springer, 2013, 864 p.
11. M. Rintala, J. Jokinen, *Olioiden ohjelmointi C++:lla*, Talentum, 2005, 466 s.
12. T. Salmi, K. Kuula, *Rakenteiden Mekaniikka*, Pressus Oy, Tampere, 2012, 464 s.
13. B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*, Addison-Wesley, 1998, 639 p.
14. STAFRA-3D -manuaali, Insinööritoimisto Lujuustekniikka Oy, 1980, 57 s.
15. K. Väänänen, Käyttäjäkokemuksen perusteet -kurssin luentomoniste, Tampereen teknillinen yliopisto, 2015, 68 s.

Liite A: M

Lopuksi muutamia pohjaa päivittän