

**Aleksi Riihiaho**

CAD-ohjelmaan liitetyn kehärakenteiden 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 kehärakenteiden 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

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](#_Toc452311295)

[2. Kehärakenteiden elementtimenetelmä 3](#_Toc452311296)

[2.1 Tekninen taivutusteoria 3](#_Toc452311297)

[2.2 Laskentamalli 4](#_Toc452311298)

[2.2.1 Elementit 5](#_Toc452311299)

[2.2.2 Koordinaatistot 5](#_Toc452311300)

[2.2.3 Solmumittausjärjestelmän kierto 6](#_Toc452311301)

[2.2.4 Ratkaisukaavat 7](#_Toc452311302)

[2.3 Tietokoneavusteinen laskenta 8](#_Toc452311303)

[2.3.1 Aktiivisarakeratkaisija 8](#_Toc452311304)

[3. Käyttöliittymäsuunnittelun periaatteita 9](#_Toc452311305)

[3.1 Käyttäjäkokemus 9](#_Toc452311306)

[3.2 Käyttäjäkeskeinen suunnittelu 10](#_Toc452311307)

[3.2.1 Normanin suunnitteluperiaatteet 11](#_Toc452311308)

[3.2.2 Nielsenin heuristiikat 12](#_Toc452311309)

[3.2.3 Shneidermanin kahdeksan kultaista sääntöä 13](#_Toc452311310)

[4. Graafisen käyttöliittymän ohjelmointi Windows-ympäristössä 14](#_Toc452311311)

[4.1 Olio-ohjelmointi 14](#_Toc452311312)

[4.1.1 Oliot ja luokat 14](#_Toc452311313)

[4.1.2 Periytyminen 16](#_Toc452311314)

[4.2 Suunnittelumallit 17](#_Toc452311315)

[4.2.1 MVC-arkkitehtuuri 17](#_Toc452311316)

[4.2.2 Visitor pattern -suunnittelumalli 18](#_Toc452311317)

[4.3 Windows Ribbon Framework 20](#_Toc452311318)

[4.4 OpenGL-piirtotyökalut 20](#_Toc452311319)

[5. Vertexin lujuuslaskentaominaisuuksien uudistaminen 21](#_Toc452311320)

[5.1 Asiakasvaatimukset 21](#_Toc452311321)

[5.2 Uudet lujuuslaskentamoduulit 23](#_Toc452311322)

[5.2.1 STAFRA-laskentamoottori 24](#_Toc452311323)

[5.2.2 Tilavuusmallien FEM-analyysi 24](#_Toc452311324)

[5.2.3 Kehärakenteiden FEM-analyysi 24](#_Toc452311325)

[5.2.4 Kehärakenteiden mitoitus 24](#_Toc452311326)

[6. Kehärakenteiden lujuuslaskentamoduulin Käyttöliittymä 25](#_Toc452311327)

[6.1 Tietorakenne 25](#_Toc452311328)

[6.2 FEA-tila 25](#_Toc452311329)

[6.2.1 Tutkimuspuu 25](#_Toc452311330)

[6.2.2 Ribbon-valikko 25](#_Toc452311331)

[6.3 Käyttötapaukset 26](#_Toc452311332)

[6.3.1 Kehän lujuustarkastelu 26](#_Toc452311333)

[6.3.2 CFS-aukkopalkin mitoitus? 26](#_Toc452311334)

[7. Jatkokehitystarpeet 27](#_Toc452311335)

[8. Yhteenveto 28](#_Toc452311336)

[Lähteet 29](#_Toc452311337)

LIITE A:

LYHENTEET JA MERKINNÄT

C++ olio-ohjelmointikieli

CAD engl. Computer-Aided Design, tietokoneavusteinen suunnittelu

CFS engl. Cold-formed steel, kylmämuovattu teräs

FEM engl. Finite element method, elementtimenetelmä

MVC engl. model-view-controller, käyttöliittymien suunnittelussa yleinen ohjelmistoarkkitehtuurityyli

STAFRA engl. Static analysis of frames, Lujuustekniikka Oy:n (nykyisin Vertex Systems Oy) kehittämä avaruuskehien statiikan laskentamoottori

# Johdanto

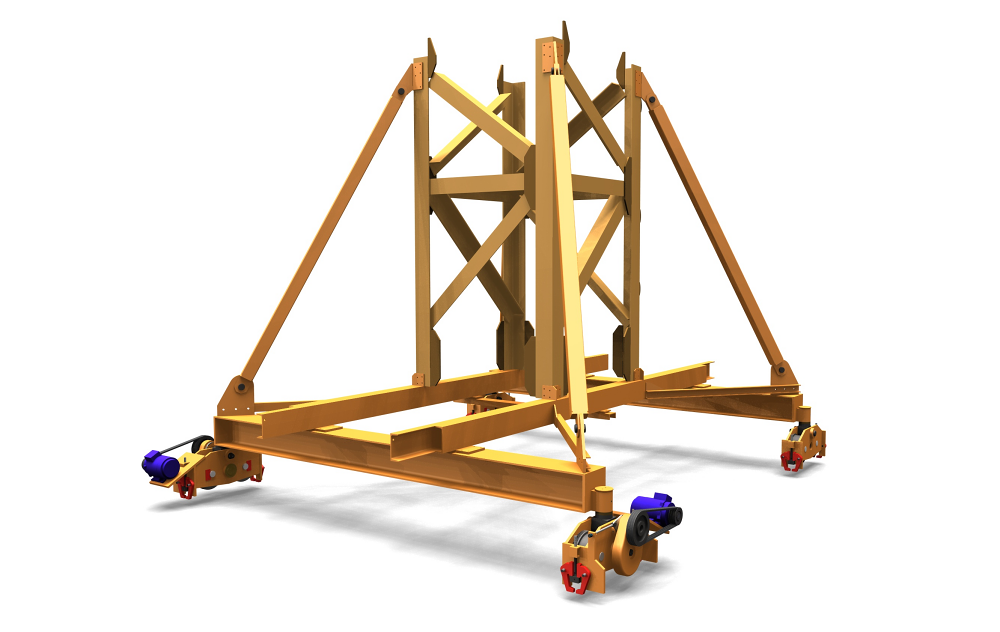
Tässä diplomityössä suunnitellaan ja toteutetaan Vertex CAD-ohjelmistoihin liitettävän kehärakenteiden lujuuslaskentamoduulin käyttöliittymä. Laskentamoduulin perustana toimii elementtimenetelmää hyödyntävä laskentamoottori. Aiemmin Vertex CAD-ohjelmistoissa kyseistä laskentamoottoria on käytetty automaattiseen ristikkorakenteiden standardin mukaiseen mitoittamiseen. Tässä työssä toteutettavan uuden käyttöliittymän tarkoituksena on laajentaa laskentamoottorin käyttömahdollisuuksia kaikkiin profiilirakenteisiin rajoittumatta mihinkään tiettyyn rakennetyyppiin.

Vertex Systems Oy on vuonna 1977 perustettu suomalainen ohjelmistoyritys. Vertex Systems Oy valmistaa ohjelmistoratkaisuja teollisuuden tarpeisiin monille eri toimijoille, joista tärkeimpinä mainittakoon esimerkiksi metalliteollisuuden kone- ja laitevalmistajat, teolliset talonrakentajat, kalusteiden valmistajat, laitostoimittajat, prosessiteollisuus sekä suunnittelutoimistot. Vertex-ohjelmistoilla on yhteensä noin 18 000 käyttäjää 37 eri maassa. [17]

Tässä diplomityössä toteutettava käyttöliittymä tullaan liittämään kolmeen eri Vertex-ohjelmistotuotteeseen:

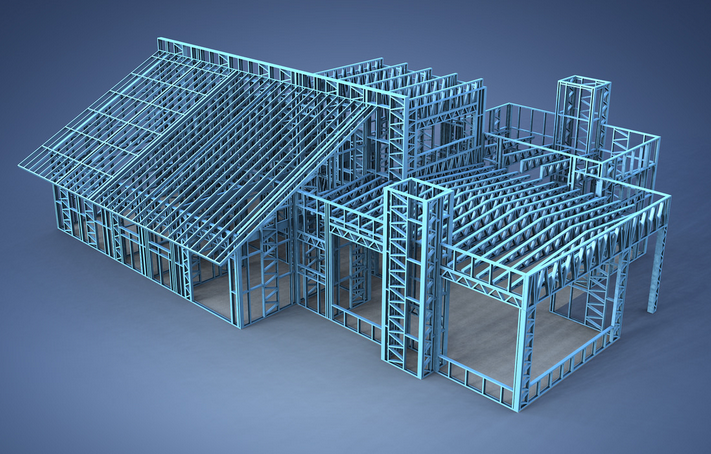
* Vertex G4 -mekaniikkasuunnitteluohjelmistoon
* Vertex G4 Plant -laitos- ja putkistosuunnitteluohjelmistoon
* Vertex BD -rakennussuunnitteluohjelmistoon

Kaikkia näitä kolmea ohjelmistoa yhdistävä tekijä on se, että niillä suunnitellaan erilaisia profiilirakenteita, kuten esimerkiksi konerunkoja, porrastorneja, hoitotasoja, ristikkorakenteita sekä rakennusten runkoja. Kuva 2 on torninosturin alavaunu, joka on tyypillinen esimerkki Vertex G4 -ohjelmistolla suunniteltavasta profiilirakenteesta.



1. Esimerkki tyypillisestä Vertex G4 -ohjelmistolla suunnitellusta profiilirakenteesta. [17]

Vertex BD -rakennussuunnitteluohjelmiston yksi suurimmista asiakaskohderyhmistä on kylmämuovatusta teräsprofiilista valmistettavien rakennusten suunnittelijat. CFS-profiileista (*Cold-formed steel*) valmistettavia rakennuksia suunnitellaan Vertex BD -ohjelmistolla laajalti ympäri maailmaa, mutta erityisesti Australiassa ja Yhdysvalloissa.



1. Esimerkki tyypillisestä Vertex BD -ohjelmistolla suunnitellusta CFS-profiilirakenteesta. [17]

Miksi tehdään sisäänrakennettu FEM-laskentaosio Vertexin ohjelmistoihin?

Minkälaisia ongelmia on tarkoitus laskea? Asiakastarve

Käyttöliittymän suunnittelu ja toteutus FEM-laskentaosiolle

Suunnittelija kohtaa työssään usein tilanteita, joissa suunniteltavalle rakenteelle tulisi tehdä lujuustarkasteluja tietyillä kuormituksilla ja tuentareunaehdoilla. Lujuustarkastelujen avulla saadaan selville rakenteen käyttäytyminen kuormituksen alaisena, jonka perusteella voidaan tehdä mitoitusta rakenneosille. Rakenteen käyttäytymistä kuvaavia suureita profiilirakenteille ovat esimerkiksi palkkien

* siirtymät,
* leikkausvoimat ja taivutusmomentit poikkileikkauksen pääsuunnissa,
* normaalivoimat sekä
* akselin suuntaiset normaalijännitykset.

# Kehärakenteiden elementtimenetelmä

Elementtimenetelmä (*Finite element method*, FEM) on matemaattinen menetelmä differentiaali- ja osittaisdifferentiaaliyhtälöiden reuna-arvotehtävien numeeriseen ratkaisemiseen. Elementtimenetelmällä voidaan muuntaa osittaisdifferentiaaliyhtälö algebralliseksi yhtälösysteemiksi, joka on helposti ratkaistavissa. Muunnos tapahtuu diskretoimalla kenttäfunktion äärettömän suuri tuntematon arvojoukko äärelliseksi määräksi solmuarvoja. Diskretoinnin jälkeen solmuarvojen ratkaisu saadaan yhtälösysteemistä. Eri tekniikan aloilla tämänlaisia tehtäviä joudutaan ratkaisemaan paljon, joten elementtimenetelmä onkin nykyään levinnyt muun muassa mekaniikan, termodynamiikan, virtausmekaniikan, murtumismekaniikan, akustiikan ja sähkötekniikan aloille. [9][14]

EPÄLIN, LIN?

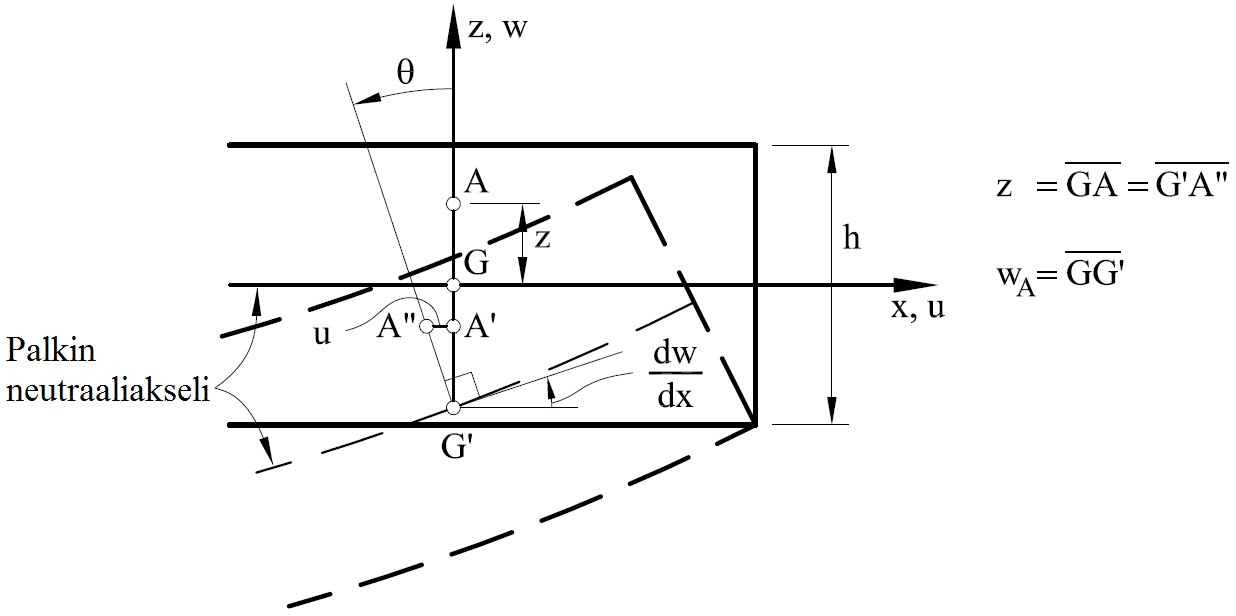
Kehärakenteiden elementtimenetelmässä rajoitutaan ratkaisemaan yksiulotteisista perusrakenneosista koostuvia rakenteita. Tällaisia rakenneosia ovat esimerkiksi sauvat, palkit ja pilarit. Tämän johdosta kehärakenteiden elementtimenetelmä voidaan perustaa suoraan rakenteiden mekaniikan analyyttisten ratkaisumenetelmien teorialle. Tämä tarkoittaa myös sitä, että kehärakenteiden elementtimenetelmä johtaa teknisen taivutusteorian puitteissa tarkkaan ratkaisuun diskretoidun mallin solmukohdissa. [14]

Tämä luku käsittelee kehärakenteiden elementtimenetelmään liittyvää teoriaa. Tämän teorian tietämystä tarvitaan tässä työssä käsiteltävän lujuuslaskentamoduulin käyttämisessä.

## Tekninen taivutusteoria

Teknistä taivutusteoriaa kutsutaan joskus myös klassiseksi palkkiteoriaksi tai sen kehittäjiensä mukaan Euler-Bernoullin palkkiteoriaksi. Se on yksinkertaisin palkin käyttäytymistä kuvaava malli. Teknisen taivutusteorian perusoletukset ovat seuraavat: [9]

1. Palkin poikkileikkaus säilyy tasona taivutuksessa.
2. Palkin poikkileikkaus ei veny taivutuksessa.
3. Palkin poikittainen leikkausmuodonmuutos on merkityksetön.



1. Teknisen taivutusteorian mukainen palkki. Kuva muokattu lähteestä [12].

Kuva 2 ja perusoletusten mukaisesti voidaan kirjoittaa siirtymille *u* ja *w* lausekkeet

ja (1)

, (2)

joissa on palkin neutraaliakselin pisteen *x* Z-suuntainen siirtymäkomponentti ja *θ* palkin poikkileikkauksen kiertymäkulma. Tekninen taivutusteoria olettaa palkin taipumat pieniksi, jolloin voidaan merkitä ja . Tällöin siirtymien likimääräisiksi lausekkeiksi saadaan

ja (3)

. (4)

Kolmannesta oletuksesta seuraa, että palkin poikkileikkauksen kiertymäkulma on sama kuin palkin neutraaliakselin kulmakerroin, jolloin siirtymän *u* lausekkeen voi kirjoittaa seuraavasti:

. (5)

## Laskentamalli

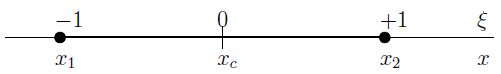
Rakenteiden mekaniikassa elementtimenetelmän perusajatuksena on jatkuvan materiaalikontinuumin diskretisoiminen äärelliseen määrään elementtejä, jotka liittyvät toisiinsa niin sanotuissa solmukohdissa. Rakenteen muodostamaa elementtijoukkoa kutsutaan laskentamalliksi. [14]

### Elementit

Elementtityypin määrittelyyn vaaditaan seuraavat asiat: [9]

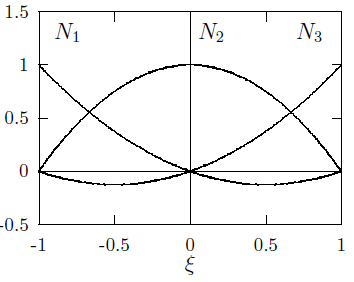
* Elementin geometria
* Muotofunktiot
* Vapausasteiden määrittely

Elementin geometria voi olla lähes mikä tahansa geometrinen kuvio tai kappale, esimerkiksi jana, nelikulmio tai tetraedri. Kehärakenteiden elementtimenetelmässä elementtien geometria on yksiulotteinen, jolloin rajoitutaan janasta tai suljetusta käyrästä muodostuviin elementteihin. Kuva 3 on esitetty yksinkertainen janaelementti, jonka paikallinen koordinaatti määritellään välillä .



1. Yksiulotteinen janaelementti. [9]

Elementtimenetelmällä muodostetun yhtälöryhmän ratkaisu antaa tuloksia vain laskentamallin solmuissa. Jotta saataisiin tuloksia myös solmujen välillä, eli elementeillä, täytyy elementeille olla määritelty muotofunktiot. Muotofunktiot interpoloivat haluttua suuretta elementin alueella. Tavallisesti muotofunktioina käytetään yksinkertaisia polynomeja. on esitetty paraboliset Lagrangen muotofunktiot, jotka ovat esimerkki yleisesti elementtimenetelmässä käytetyistä muotofunktioista. [9]

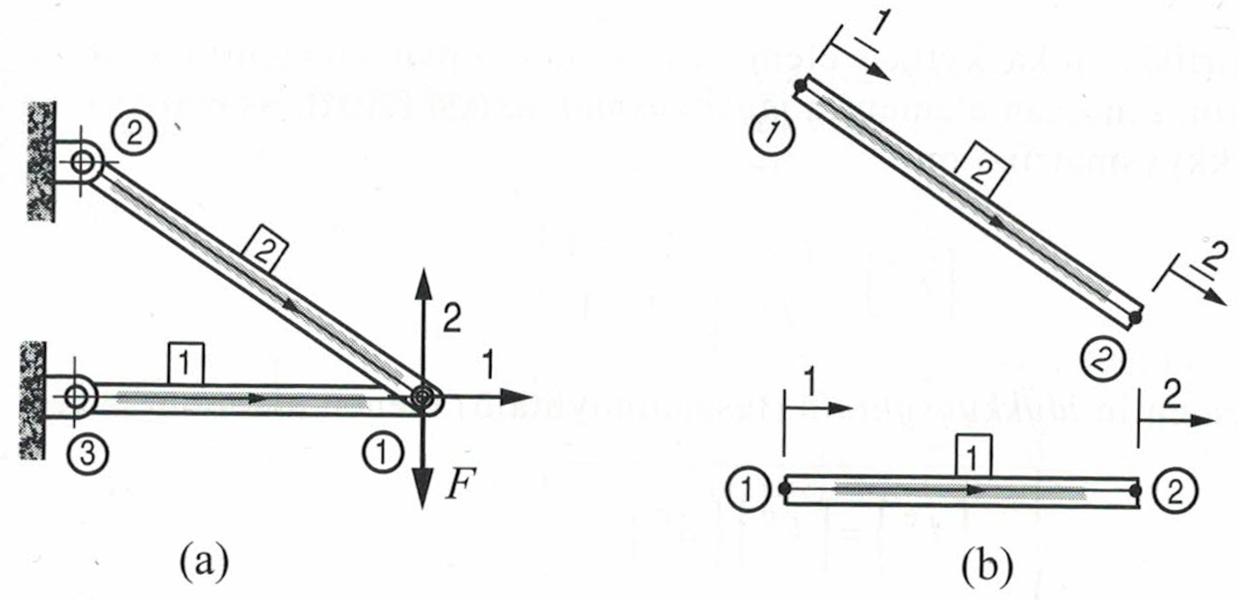


1. Paraboliset Lagrangen muotofunktiot. Kuva muokattu lähteestä [9].

### Koordinaatistot

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

Paikalliskoordinaatistolla tarkoitetaan yksittäisen elementin sisäistä 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 rakennekoordinaatisto. Solmumittausjärjestelmällä tarkoitetaan kussakin tilanteessa voimassa olevaa koordinaatistoa, jossa solmumittaus suoritetaan.



1. Laskentamalli (a) ja yksittäiset elementit (b). Kuva muokattu lähteestä [14].

Kuva 5 vasemmalla on esitetty yksinkertaisen sauvarakenteen laskentamalli. Laskentamallin solmumittaus rakennekoordinaatistossa on merkitty nuolilla 1 ja 2. Oikean puoleisessa kuvassa on laskentamallin yksittäiset elementit ja niiden paikalliset solmumittausjärjestelmät. Laskentamallia luodessa yksittäisten elementtien paikalliset solmumittausjärjestelmät tulee muuttaa rakenteen solmumittausjärjestelmän mukaiseksi. Muutos toteutetaan koordinaatistoa kiertämällä.

### Solmumittausjärjestelmän kierto

Palkkielementtien solmumittausjärjestelmää joudutaan usein kiertämään, koska palkit voivat osoittaa rakennekoordinaatistossa mielivaltaiseen suuntaan. Jotta sijoittelusummaus voidaan suorittaa, tulee koko rakenteessa olevien yksittäisten palkkielementtien solmumittausten olla samansuuntaiset. Koordinaatiston kierto elementin paikalliskoordinaatistosta rakennekoordinaatistoon ja toisin päin tehdään niin sanotulla koordinaatiston kiertomatriisilla. Kiertomatriisi kootaan siten, että paikalliskoordinaatiston kantavektoreilla **i**, **j** ja **k** lausutun vektorin **v** sekä rakennekoordinaatiston kantavektoreilla **i**, **j** ja **k** lausutun vektorin **v** välillä on lineaarinen yhteys:

. (6)

Esimerkiksi koordinaatistoa kulman *θ* verran X-akselin ympäri vastapäivään kääntävä kiertomatriisi *T* voidaan kirjoittaa seuraavasti:

. (7)

### Ratkaisukaavat

Yksittäisen elementin tasapainoyhtälöistä on johdettavissa kerroinmatriisi, joka kytkee elementin solmuille kohdistuvat voimat vastaaviksi solmusiirtymiksi. Tätä kerroinmatriisia kutsutaan elementin jäykkyysmatriisiksi. Koko laskentamallin jäykkyyttä kuvaa rakenteen jäykkyysmatriisi. Se kootaan laskentamallin muodostavien elementtien yksittäisistä jäykkyysmatriiseista sijoittelusummauksella. Sijoittelusummauksessa yksittäisten elementtien jäykkyysmatriisien alkiot sijoitetaan rakenteen jäykkyysmatriisin vastaaville paikoille rakenteen solmunumeroinnin mukaisesti. Eri elementeistä tulevat samaan rakennesolmuun liittyvät alkiot summataan yhteen. Sijoittelusummaus edellyttää, että yksittäisten elementtien paikallinen solmumittausjärjestelmä on yhdensuuntainen rakenteen solmumittausjärjestelmän kanssa. Sijoittelusummaus on helposti ohjelmoitavissa ja tehokas tapa koota rakenteen jäykkyysmatriisi tietokonelaskennan kannalta. [14]

Rakenteen laskentamallin kuormitukset kootaan rakennesolmujen kuormitusvektoriin

, (8)

jossa on suoraan rakennesolmuille annettu ulkoinen solmukuormitusvektori ja rakennesolmujen ekvivalenttinen solmukuormitusvektori. Rakenteeseen kohdistuva ulkoinen kuormitus jakaantuu usein elementin alueelle, eikä suoraan solmuille, jolloin nämä kenttäkuormitukset täytyy muuntaa solmuvoimiksi. Solmuvoimiksi muunnetut kenttäkuormitukset muodostavat ekvivalenttisen solmukuormitusvektorin. Yksittäisten elementtien ekvivalenttisista solmukuormitusvektoreista saadaan koottua rakennesolmujen ekvivalenttinen solmukuormitusvektori sijoittelusummauksella.

Rakenteen laskentamallin siirtymät saadaan ratkaistua rakennesolmujen tasapainoyhtälöstä

, (9)

jossa on rakenteen jäykkyysmatriisi ja rakennesolmujen siirtymävektori.

## Tietokoneavusteinen laskenta

Matriisien tallennus skyline-muotoon

### Aktiivisarakeratkaisija

Suurille matriiseille

# Käyttöliittymäsuunnittelun periaatteita

Ohjelmistoyrityksen kannattaa panostaa tuotteidensa hyvään ja helppoon käytettävyyteen, sillä ne vaikuttavat suoraan tuotteiden laatuun ja vähentävät monia kustannuksia. Ottamalla huomioon käytettävyys jo tuotekehitysprojektin alkuvaiheessa saadaan pienennettyä kehityskustannuksia. Kun tiedetään jo projektin alkuvaiheessa minkälainen käyttöliittymä tyydyttää käyttäjien tarpeet, kehitystyöhön kuluva aika ja kustannukset pienevät. Myös tuotteen ylläpitokustannukset saadaan pienemmiksi, kun ohjelmistoa on helppo käyttää, sillä silloin asiakastuen tarve on vähäisempi.

Tässä luvussa perehdytään käyttäjäkokemuksen määritelmään sekä käyttäjäkeskeisen suunnittelun perusperiaatteisiin. Luvun lopussa esitellään muutamia yleisesti hyväksi havaittuja suunnitteluperiaatteita hyvän käytettävyyden edistämiseksi erityisesti ohjelmistojen käyttöliittymien näkökulmasta.

## 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”*. [18]

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, joilla tuotteen määritellyt käyttäjät saavuttavat 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. [18]

Tuotteen käyttöympäristö vaikuttaa merkittävästi käyttäjäkokemuksen syntymiseen. Käyttöympäristö, eli käyttökonteksti, määrittelee millaisessa 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, millaisilla laitteilla loppukäyttäjät tulevat tuotetta käyttämään. Sosiaalisia käyttöympäristöjä ovat esimerkiksi koti ja työpaikka. Kotona tunnelma on usein rentoutunut kun taas työpaikalla ilmapiiri voi olla virallisempi. [18]



1. Käyttäjäkokemuksen syntyminen käyttäjän ja tuotteen välisessä vuorovaikutuksessa [18]

Kuva 6 on havainnollistettu käyttäjäkokemuksen syntymistä käyttäjän ja järjestelmän vuorovaikutuksen tuloksena. Kuvasta korostuu se, että käyttäjäkokemuksen syntyminen ei ole pelkästään seurausta käytettävyydestä, vaan siihen vaikuttaa oleellisesti myös itse käyttäjä sekä käyttöympäristö.

## Käyttäjäkeskeinen suunnittelu

Käyttäjäkeskeinen suunnittelu on prosessi, jossa tuotteen hyvää käyttäjäkokemusta suunnitellaan hyödyntäen olemassa olevaa käyttäjätietoa. Oikeilta tuotteen loppukäyttäjiltä saatu tieto on usein arvokkaampaa ja oikeellisempaa kuin käyttöliittymää suunnittelevan ohjelmistokehittäjän intuitio. Käyttäjäkeskeisessä suunnittelussa pyritään ymmärtämään ihmisen käyttäytymistä ja taitoja sekä hyödyntämään tätä ymmärrystä käyttöliittymän suunnittelussa.

Ohjelmistoprojekteissa yleensäkin tärkeä asia asiakaslähtöisyys on erityisen tärkeää käyttäjäkeskeisessä suunnittelussa. Asiakas ja sen tarpeet on tunnettava riittävän hyvin, jotta voidaan suunnitella tuotteen käyttäjäkokemusta oikealle käyttäjäryhmälle. Ohjelmistoprojekteissa asiakkaan tulisikin olla mukana suunnittelussa jo tuotteen vaatimusten määrittelyvaiheessa. Vaatimusten määrittelyssä käyttäjien tarpeiden kartoittaminen koostuu haastatteluista, käyttäjien tarkkailusta sekä kilpailija-analyyseistä. [5]

Käyttäjäkeskeinen suunnittelu on usein luonteeltaan iteratiivista. Asiakkailta ja loppukäyttäjiltä haetaan jo suunnitteluprosessin aikana toistuvasti palautetta siihen asti toteutetuista ratkaisuista. Tarpeeksi ajoissa saatu palaute helpottaa viemään järjestelmää käyttäjäystävällisempään suuntaan ja ongelmakohtien korjaaminen on huomattavasti halvempaa suunnitteluvaiheessa kuin vasta järjestelmän valmistuttua. Iteratiivisen suunnittelun haittapuolena on vaikeus arvioida suunnittelun aikataulua. Suunnittelu- ja toteutuskierroksia saattaa tulla useita ennen kuin järjestelmä saadaan käyttäjää tyydyttävälle tasolle. [5]

Eräs tehokas käyttäjäkeskeisen suunnittelun työkalu on prototypointi. Toteutettavasta tuotteesta tehty konkreettinen hahmotelma paljastaa käytettävyysongelmat paljon paremmin kuin pelkkä paperille kirjoitettu tuotemäärittely. Prototypointi on halpaa ja nopeaa sekä se auttaa luomaan yhteisen kielen suunnittelijoiden ja käyttäjien välille, kun konkreettista tuotetta pääsee kokeilemaan.

Ohjelmistotuotannon alalla paperiprototyyppi on yleisesti käytetty keino testata järjestelmän käytettävyyttä suunnittelun alkuvaiheessa ennen kuin edes ohjelmointityötä on aloitettu. Paperiprototyyppi voi olla esimerkiksi joukko paperille tulostettuja kuvia tai PowerPoint-esitys, joka mallintaa suunniteltavan järjestelmän käyttöä. Paperiprototypoinnin päämääränä on käyttöliittymän eri näkymien ja niiden välisen vuorovaikutuksen nopea testaaminen. Paperiprototypointi on nopea ja halpa keino testata erilaisia ratkaisuvaihtoehtoja järjestelmän käyttöliittymälle aivan suunnittelun alkuvaiheessa.

Seuraavissa luvuissa 3.3.1 – 3.3.3 käsitellään erilaisia hyvän käyttäjäkeskeisen suunnittelun periaatteita, jotka ovat sovellettavissa käyttöliittymäsuunnitteluun. Suunnitteluperiaatteet soveltuvat myös käytettävyyden heuristiseen arviointiin. [18]

### Normanin suunnitteluperiaatteet

Käyttäjäkeskeisen suunnittelun alalla uraa tehnyt professori Donald A. Norman kertoo kirjassaan ”The Design of Everyday Things” (2013) [11] 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

Kun ihminen muodostaa käsitystä tuotteen toiminnasta, hän tekee näkemästään rakenteesta tulkintoja rakenteen 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) [10] 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) [15] esitellään kahdeksan kultaista sääntöä erityisesti ohjelmistojen käyttöliittymien dialogien suunnitteluun ja arviointiin. Säännöt ovat kuitenkin sovellettavissa ohjenuoriksi myös koko järjestelmän suunnitteluun [18]. 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.

## Olio-ohjelmointi

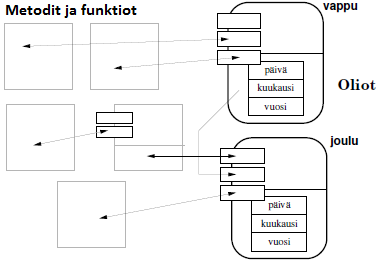
Ohjelmistotekniikan yhtenä vaikeimpana ongelmana pidetään suurten ohjelmistojen tekemisen monimutkaisuutta. Tietokonelaitteistojen tekniikoiden räjähdysmäinen kehitys on mahdollistanut entistä suurempien ja kompleksisempien ohjelmistojen suorittamisen. Ohjelmistojen tekemiseen tarvittavat työkalut ja 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. [13]

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, perusperiaatteina ovat 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. [13]

### 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. [13]

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. [13]



1. Oliot ja niiden tietorakenne sekä julkinen rajapinta. Kuva muokattu lähteestä [13].

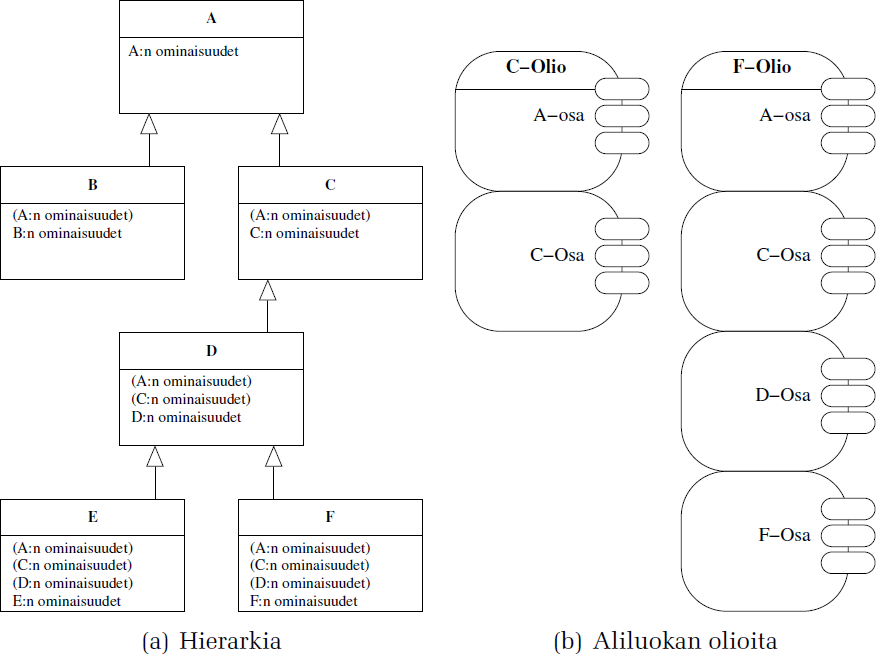
Kuva 7 on havainnollistettu tyypillisiä olio-ohjelmoinnissa käytettäviä olioita. Oliot ”vappu” ja ”joulu” ovat samasta luokasta tehtyjä instansseja, ja niiden tarkoitus on esittää päiväystä. Päiväystä esittävän olion tietorakenteeseen kuuluu siis tässä tapauksessa päivä, kuukausi ja vuosi, jotka voidaan 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. [13]

### 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. [13]

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 [8].

Kuva 8 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

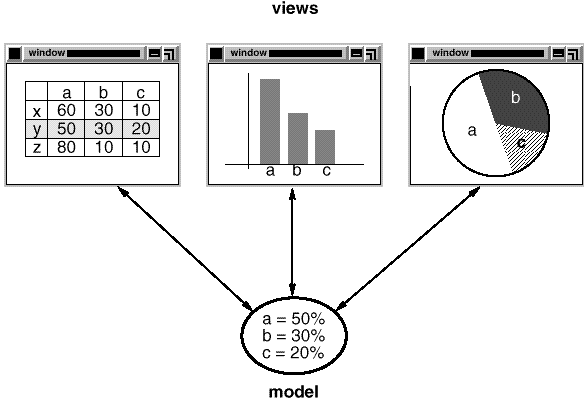
Olio-ohjelmoinnissa ohjelmiston toiminnallisuus muodostuu suuren oliojoukon yhteistoiminnan vaikutuksesta. Hyvin suunniteltu ohjelmisto sisältää sellaisia oliorakenteita, jotka ovat uudelleenkäytettäviä myös muissa kohteissa. Tällä tarkoitetaan suunnittelun geneerisyyttä. Kirjassa *”Design Patterns: Elements of Reusable Object-Oriented Software”* [4] on kerätty hyviksi havaittuja oliosuunnittelun käytäntöjä erilaisiin ohjelmointiongelmiin. Näistä käytännöistä käytetään kirjassa nimitystä suunnittelumalli. Kaikki kirjan käsittelemät suunnittelumallit ovat sellaisia, jotka on havaittu toimiviksi kahden tai useamman toisistaan riippumattoman ohjelmistoprojektihenkilöstön toimesta.

Seuraavissa aliluvuissa esitellään useammasta suunnittelumallista koostuvaa MVC-arkkitehtuuria sekä ”Visitor pattern” -suunnittelumallia. Niiden periaatteita on sovellettu tässä työssä toteutetussa käyttöliittymässä.

### MVC-arkkitehtuuri

MVC-arkkitehtuuri (*Model-view-controller)* on kolmesta erilaisesta luokasta koostuva ohjelmistoarkkitehtuurityyli. Se on yleinen suunnitteluperiaate monien käyttöliittymien toteutuksissa. MVC-arkkitehtuurin luokat ovat malli (model), näkymä (view) ja käsittelijä (controller). Sen etuna on käyttöliittymän näkyvän osan toteutuksen erottaminen sen esittämän tietomallin rakenteesta. Tällöin mallin tieto ei ole riippuvainen näkymästä, vaan saman mallin sisältö voidaan esittää usean erilaisen näkymän kautta. Malli ei ole riippuvainen myöskään käsittelijästä, joten se voidaan suunnitella täysin irrallaan käsittelijästä. Käsittelijä toimii käyttäjältä tulevien komentojen vastaanottajana muuntaen mallia ja näkymää niiden perusteella. [4]

Kuva 9 on havainnollistettu tyypillisen MVC-arkkitehtuurin rakennetta. Se sisältää kolmesta muuttujasta a, b ja c koostuvan mallin sekä kolme erilaista näkymään siihen. Yksinkertaistuksen vuoksi käsittelijä on jätetty kuvasta pois. Taulukko, histogrammi ja ympyräkaavio esittävät mallin tiedon eri tavoin. Malli viestii näkymille, kun siihen tulee muutoksia. Näkymät viestivät mallin kanssa saadakseen pääsyn näytettävään tietoon. Tällä tavoin malli ja näkymät pysyvät aina ajan tasalla. Käsittelijä, jota kuvaan ei ole piirretty, voisi toimia esimerkiksi näppäimistön ja hiiren painallusten tulkitsijana. Koska käsittelijä toteutetaan MVC-arkkitehtuurissa omana luokkanaan, se on helppo vaihtaa toiseen esimerkiksi hieman erilaiseen toteutukseen jotain toista käyttötarkoitusta varten.



1. MVC-arkkitehtuurin malli ja näkymät. [4]

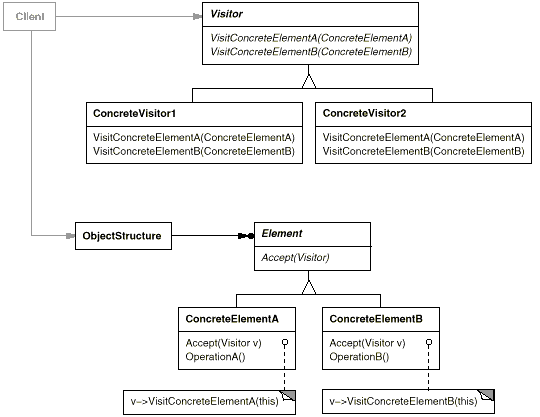
MVC-arkkitehtuuri koostuu itse asiassa kolmen erilaisen suunnittelumallin hyödyntämisestä:

* Observer-suunnittelumallista, jossa muutos yhden luokan oliossa voi vaikuttaa kuinka moneen tahansa muuhun olioon tarvitsematta mitään tietoa muiden olioiden toteutusyksityiskohdista. Esimerkiksi Kuva 9 tietomallin suhde näkymiin voi olla tällainen.
* Composite-suunnittelumallista, jossa yksittäisiä olioita käsitellään samalla tavoin kuin koosteolioita. Koosteolioilla on yhteinen kantaluokka yksittäisten olioiden kanssa, ja ne sisältävät ryhmän yksittäisiä olioita. Esimerkiksi kuvan 8 näkymät voisivat olla koosteolioita, jolloin niiden sisällä voisi olla vielä muita näkymiä.
* Strategy-suunnittelumallista, jossa jokin algoritmi toteutetaan luokkana. MVC-arkkitehtuurin tapauksessa algoritmina toimii käsittelijä-luokka, joka huolehtii käyttäjän antamien komentojen toteuttamisesta. Käsittelijä-luokkia voi olla useita ja niitä on mahdollista vaihtaa jopa ohjelman ajon aikana. Tällä tavoin saadaan muutettua näkymän vastetta käyttäjän syötteisiin helposti ainoastaan käsittelijä-oliota vaihtamalla.

### Visitor pattern -suunnittelumalli

Visitor pattern -suunnittelumalli mahdollistaa tietorakenteen ja siihen liittyvien operaatioiden irrottamisen toisistaan. Suurin käytännön hyöty tästä on se, että uusien tietorakennetta läpikäyvien operaatioiden toteuttaminen onnistuu helposti muuttamatta itse tietorakennetta. Visitor pattern -malli on hyödyllisin silloin, kun tietorakenteen tiedetään pysyvän kohtuullisen muuttumattomana, mutta tietorakenteeseen liittyviä uusia operaatioita halutaan mahdollisesti toteuttaa myöhemmin lisää. Se on erityisen käyttökelpoinen myös silloin, kun tietorakenne koostuu useista erilaiset rajapinnat sisältävistä luokista, joille kaikille halutaan tehdä saman tyyppisiä, mutta hieman toisistaan poikkeavia operaatioita. Operaatioita ei tällöin tarvitse toteuttaa tietorakenteen sisälle, vaan ne voidaan ulkoistaa omiksi luokikseen.

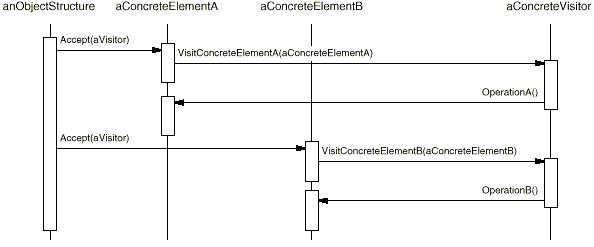
Visitor pattern -suunnittelumallin perusidea on se, että tietorakenne itse huolehtii alkioidensa läpikäymisen, ja erillinen Visitor-luokka suorittaa halutun operaation yhdelle tietorakenteen alkiolle. Kuva 10 on esitetty luokkakaavio, joka havainnollistaa suunnitteluperiaatteen toimintaa.



1. Visitor pattern -suunnittelumalli. [4]

Visitor pattern -suunnittelumallin voidaan katsoa koostuvan seuraavista viidestä komponenttityypistä:

* **Visitor -luokka,** joka esittelee vastaavan visit-operaation jokaiselle yksittäiselle tietorakenteen alkion luokalle. Visitor-luokka toimii kantaluokkana ConcreteVisitor-luokille.
* **ConcreteVisitor -luokka,** joka toteuttaa suoritettavan operaation yksittäiselle tietorakenteen alkiolle. Jokaista tietorakenteen alkion luokkaa kohden on myös oma funktionsa, joiden toteutus voi olla erilainen. ConcreteVisitor-luokan olio voi myös säilöä tietoa itseensä käydessään läpi tietorakennetta.
* **Element -luokka,** joka esittelee accept-operaation. Accept-operaatio saa parametrinaan Visitor-tyyppisen olion. Element-luokka toimii kantaluokkana kaikille tietorakenteen alkioiden luokille.
* **ConcreteElement -luokka,** joka **v**astaa yksittäisen tietorakenteen alkion luokkaa. Tämä luokka toteuttaa accept-operaation, joka ikään kuin päästää Visitor-tyyppisen olion tekemään operaationsa ConcreteElement-tyyppiseen olioon.
* **Tietorakenne,** joka koostuu Element-luokasta periytetyistä olioista. Siihen kohdistuvat operaatiot ovat Visitor-luokasta periytettyjen ConcreteVisitor-olioiden toteuttamia. Tietorakenteen muoto on täysin vapaa. Se voi olla esimerkiksi lista, puu tai graafi.



1. Visitor pattern -suunnittelumallin mukainen tietorakenteen läpikäynti. [4]

Kuva 11 havainnollistaa Visitor pattern -suunnittelumallin mukaista tietorakenteen läpikäyntiä. Kuvan tietorakenne on yksinkertaistettu siten, että se koostuu vain kahdesta alkiosta ConcreteElementA ja ConcreteElementB. Tietorakenteen alkiot kutsuvat Accept-funktiossa parametrina saadun Visitor-tyyppisen olion visit-funktiota. Visit-funktio taas saa parametrikseen käsiteltävänä olevan tietorakenteen alkion, jolle Visitor-olio suorittaa itse operaation.

## Windows Ribbon Framework

## OpenGL-piirtotyökalut

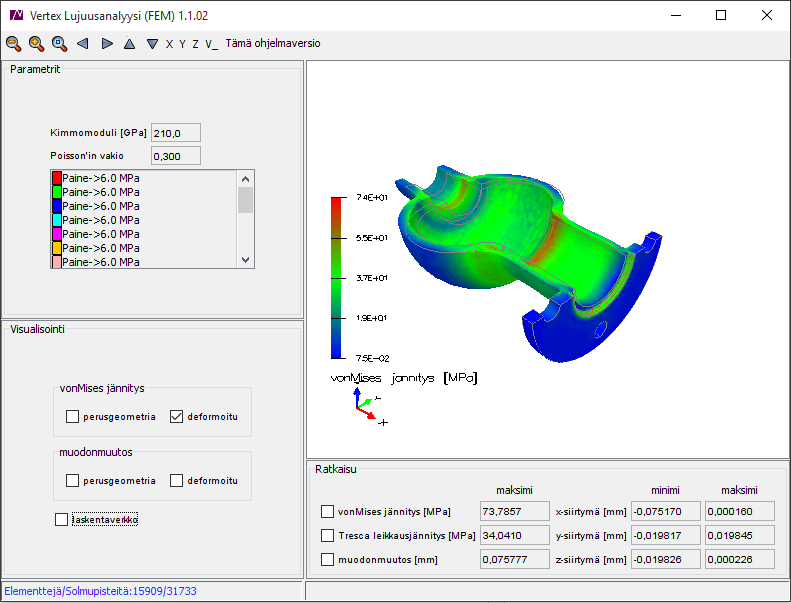
# Vertexin lujuuslaskentaominaisuuksien uudistaminen

Vertexillä ryhdyttiin vuoden 2015 lopussa projektiin, jonka tavoitteena oli kehittää Vertex-ohjelmistojen lujuuslaskentaominaisuuksia kilpailukykyisempään suuntaan seuraaviin vuonna 2016 marraskuussa julkaistaviin versioihin.

## Asiakasvaatimukset

Aiemmin Vertex-ohjelmistoissa on ollut seuraavia lujuuslaskentaominaisuuksia:

* Yksittäisille kappaleille soveltuva tilavuuselementtejä hyödyntävä lujuusanalyysi Vertex G4 -ohjelmassa.
* Ristikkorakenteiden tai yksittäisen palkin mitoittaminen kantavien rakenteiden suunnittelustandardien mukaisesti Vertex BD-ohjelmassa. Tuettuja standardeja ovat Australian ja Ison-Britannian suunnittelustandardit.



1. Vertex G4:n yksittäisen tilavuusmallin lujuusanalyysi.

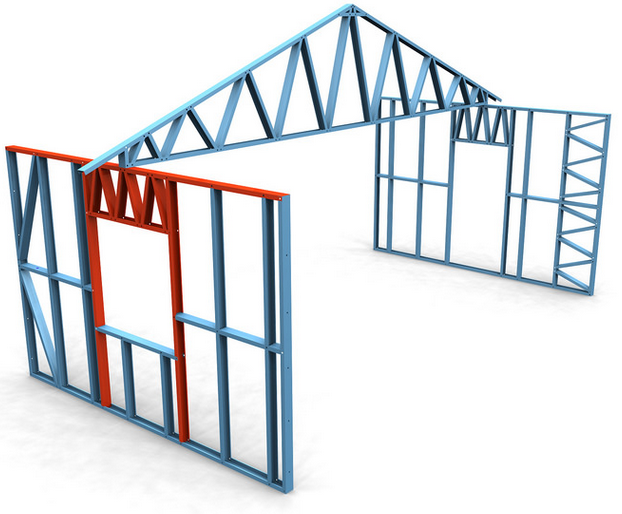
Kuva 12 näkyy Vertex G4 -ohjelmassa mukana oleva yksittäisen tilavuusmallin lujuuslaskentamoduuli. Se on kehitetty yhteistyössä laskennallisen teknologian palveluja tarjoavan Numerola Oy:n kanssa ja se pohjautuu heidän kehittämään ohjelmistokomponenttiin. Lujuuslaskentamoduulin heikkoutena on se, että sillä voidaan analysoida ainoastaan yksittäisiä kappaleita. Kokoonpanojen analysoiminen on mahdollista ainoastaan tekemällä kokoonpanomallista yksittäinen kappale boolean-operaatiolla. Tällöin kokoonpanojen kappaleiden väliset kontaktipinnat häviävät ja mahdollisuus analysoida kosketuksissa olevia kappaleita katoaa. Laskentamallin elementtiverkon luominen on myös tuottanut ongelmia ohuilla rakenteilla. Esimerkiksi joillekin putkipalkeille elementtiverkon luominen ei onnistu lainkaan.

Vertexin asiakaskunnalta kyseltiin, minkälaisia kokemuksia heillä oli nykyisestä Vertex G4 -ohjelmassa olevasta lujuuslaskentamoduulista. Asiakkailta saatiin paljon kehitysehdotuksia ja toiveita tuleviin versioihin. Alla on listattu eniten esiin nousseita toiveita:

* Kehärakenteiden 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

KUVA Vertex BD Truss engineering

KUVAVIITE näkyy Vertex BD -ohjelmassa mukana oleva ristikkorakenteiden mitoit... Tällä hetkellä Vertex BD tukee ainoastaan ristikkorakenteiden sekä yksittäisen palkin suunnittelustandardin mukaista mitoitusta. Asiakaskunnasta on tullut toiveita, että Vertex BD -ohjelmalla voisi mitoittaa myös muunlaisia rakenteita. Esimerkiksi Kuva 13 näkyvä ikkuna-aukkopalkki on tyypillinen kohde, jota asiakkaat haluaisivat mitoittaa. Myös Vertex G4 Plant -laitossuunnitteluohjelmiston asiakaskunnasta on tullut toiveita mahdollisuuteen mitoittaa suunnittelustandardien mukaisesti esimerkiksi laitosten hoitotasoja.



1. CFS-profiilirakenteinen ikkunan aukkopalkki. [17]

Vertex BD -ohjelman ristikkorakenteiden mitoitusmoduulin käyttö on tehty todella yksinkertaiseksi. Käyttäjän tarvitsee vain osoittaa malliin tulevat kuormat ja niiden tyypit. Rakenteesta luotavaa FEM-mallia ei näytetä käyttäjälle missään vaiheessa, vaan se luodaan automaattisesti. Käyttäjä voi asettaa kuormat mihin kohtaan mallia tahansa ja ne muutetaan ratkaisun aikana FEM-mallin vastaaviksi solmukuormituksiksi. Tällainen yhdelle rakennetyypille räätälöity mitoitustyökalu on todella tehokas ja helppokäyttöinen suunnittelijalle. Toisaalta, jos rakenteen FEM-mallissa on jotain korjattavaa ja laskenta ei mene läpi, käyttäjällä ei ole mitään mahdollisuuksia korjata sitä tai edes saada selville syytä, miksi FEM-laskenta ei onnistu.

Suuret ristikot ja avaruuskehät, joissa todella paljon osia.

## Uudet lujuuslaskentamoduulit

Asiakasvaatimusten pohjalta tehtiin päätös, että seuraaviin vuonna 2016 marraskuussa julkaistaviin Vertexin versioihin tullaan lisäämään kolme erilailla hinnoiteltavaa lisäoptiota:

* Tilavuusmallien FEM-analyysi Vertex G4 ja G4 Plant -ohjelmiin. Tämä laskentamoduuli tulee korvaamaan entisen Numerola Oy:n kanssa yhteistyössä tehdyn moduulin.
* Kehärakenteiden FEM-analyysi Vertex G4, G4 Plant ja BD -ohjelmiin.
* Kehärakenteiden mitoitus Vertex BD -ohjelmaan, joka vaatii myös kehärakenteiden FEM-analyysi -lisäoption.

Uudistuneet käyttöliittymät. palkkiFEM-käyttöliittymä tämän työn aiheena

Seuraavissa aliluvuissa 5.2.1 – 5.2.4 esitellään kaikkia kolmea kehitettävää lujuuslaskentamoduulia sekä niiden perustana toimivaa STAFRA-laskentamoottoria.

### STAFRA-laskentamoottori

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 kehärakenteiden 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 paikalliskoordinaatiston y- ja z-suunnissa saadaan kaavoilla

ja (10)

palkkielementin paikalliskoordinaatiston 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. [16]

STAFRA on käytössä Vertex BD:n nykyisessä kattoristikoiden mitoituslaskennassa.

Hyödynnetään tulevaisuudessa myös tilavuuselementeille (jäykkyysmatriisit, kolmoiointi, orjuutukset). Toimii pohjana solidFEM ja beamFEM

### Tilavuusmallien FEM-analyysi

STAFRA + spatial mesh + uusi käyttöliittymä.

### Kehärakenteiden FEM-analyysi

STAFRA + uusi käyttöliittymä.

### Kehärakenteiden mitoitus

Lisäoptio kehärakenteiden lujuuslaskentamoduuliin. STAFRA laskee rasitukset, DesignEngine.dll mitoittaa. Loda patterns, load combinations. Käyttöliittymässä extraa

# Kehärakenteiden lujuuslaskentamoduulin Käyttöliittymä

Tässä perehdytään palkkiFEM:iin (ja mitoitus?)

## Tietorakenne

Luokkakaavio

Datainterface: attribuuttitekstit-virtuaalinen oliomalli. Historialliset syyt takana, miksi tallennetaan osien attribuuttiteksteihin

## FEA-tila

Geometrian muokkaus estetty. Tilasta poistuminen: OK tallentaa attribuuttitekstit, Cancel hylkää muutokset.

### Tutkimuspuu

Valmiina olleet käyttöliittymäkomponentit puunäkymälle: studytree\_cont

Tutkimusten luonti, poisto. Tutkimustyypit

Tutkimusten hallinnointi: kuormien valinta, piilotus, kontekstimenu, tuennat.

Kuormakaaviot. Mihin käytetään? Load combinations

symbolit

### Ribbon-valikko

solmujen kontekstitab: Solmun lokaalikoordinaatiston asettaminen, tuentojen asettaminen, orjuutus. Esivalinnat

## Käyttötapaukset

### Kehän lujuustarkastelu

### CFS-aukkopalkin mitoitus?

# Jatkokehitystarpeet

Käyttöliittymä solidielementtien laskennalle

Ristikkorakenteiden mitoituksen FEM-mallia pystyisi muokkaamaan kehärakenteiden FEM-analyysin työkaluilla

Kuorielementit (ohutlevyosien laskenta)

Epälineaaristen ongelmien ratkaisu

Iteratiiviset ratkaisumenetelmät

Automatiikka orjuutuksiin

# 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. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns – Elements of Reusable Object-Oriented Software,* Addison-Wesley, 1995, 386 p.
5. I. Haikala, J. Märijärvi, *Ohjelmistotuotanto*, Talentum Media Oyj, 2006, 440 s.
6. R.C. Hibbeler, *Structural Analysis*, Pearson Prentice Hall, 2012, 695 p.
7. M. Hirz, W. Dietrich, A. Gfrerrer, J. Lang, *Integrated Computer-Aided Design in Automotive Development*, Springer-Verlag Berlin Heidelberg, 2013, 466 p.
8. K. Koskimies, *Oliokirja*, Talentum Oyj, 2000, 422 s.
9. R. Kouhia, M. Tuomala, *Johdatus mekaniikan ja sähkömagnetiikan numeerisiin menetelmiin*, luentomoniste, Tampereen teknillinen yliopisto, 2014, 417 s.
10. J. Nielsen, *Usability engineering,* Academic Press, 1993, 362 p.
11. D. Norman, *The Design of Everyday Things – Revised and Expanded Edition*, Basic Books, 2013, 347 p.
12. E. Oñate, *Structural Analysis with the Finite Element Method. Linear Statics. Volume 2. Beams, Plates and Shells*, Springer, 2013, 864 p.
13. M. Rintala, J. Jokinen, *Olioiden ohjelmointi C++:lla*, Talentum, 2005, 466 s.
14. T. Salmi, K. Kuula, *Rakenteiden Mekaniikka*, Pressus Oy, Tampere, 2012, 464 s.
15. B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*, Addison-Wesley, 1998, 639 p.
16. STAFRA-3D -manuaali, Insinööritoimisto Lujuustekniikka Oy, 1980, 57 s.
17. Vertex - Suunnitteluohjelmistot Tiedonhallintaohjelmistot 3D CAD PDM PLM, Vertex Systems Oy, 2016. Saatavissa (viitattu 24.5.2016): http://www.vertex.fi/web/fi/
18. K. Väänänen, Käyttäjäkokemuksen perusteet -kurssin luentomoniste, Tampereen teknillinen yliopisto, 2015, 68 s.

Liite A: M

Lopuksi muutamia pohjaa päivittän