

PREFIXIEN TARKISTUSTYÖKALU TEKLA STRUCTU- RESIIN

Saari Roosa

Opinnäytetyö
Rakennus- ja yhdyskuntatekniikka
Insinööri (AMK)

2020

Rakennus- ja yhdyskuntatekniikan koulutus
Insinööri (AMK)

Tekijä	Roosa Saari	Vuosi	2020
Ohjaaja	Heikki Ala-Louko		
Toimeksiantaja	Optiplan Oy		
Työn nimi	Prefixien tarkistustyökalu Tekla Structuresiin		
Sivu- ja liitesivumäärä	45 + 7		

Opinnäytetyössä syvennyttiin tietomallintamisen kehittämismahdollisuuksiin. Opinnäytetyön tavoitteena oli kehittää Optiplanin sisälle toimiva prefixien tarkistustyökalu Tekla Structuresiin. Toimivan työkalun avulla rakenne- ja elementtisuunnittelijan työskentely helpottuu ja nopeutuu.

Tutkimuksen tekeminen ja kehitystyöprosessi aloitettiin luomalla Tekla Structuresiin erilaisia mallinnustapauksia pilareista, palkeista, laatoista ja seinistä. Toimeksiantajalta saatua logiikaltaan toimivaa ohjelmaa testattiin luotuihin malleihin. Sellaisia tapauksia jatkokehitettiin, joita ohjelma ei kyennyt käsittelemään. Näitä olivat esimerkiksi ontelolaatta sekä teräksiset osat, joilla oli alikokoonpano prefix.

Ohjelman kehitys ja käyttöliittymän muokkaus tapahtui Visual Studiossa C#- kielellä. Ohjelmaa testattiin kehitystyöprosessin aikana Teklassa. Käyttöliittymästä tehtiin toimiva sijoittamalla painikkeet järkevästi ja lisäämällä ohjelman käyttäjälle käyttöohjeet.

Alikokoonpano prefixien kehitystyö tulee jatkumaan tulevaisuudessa Tekla- kehitysryhmässä. Valmis ohjelma logoineen ja makroineen tullaan lisäämään Teklaan .exe-tiedostona. Tiedotus ohjelman lisäyksestä Optiplanin rakennesuunnittelijoille tapahtuu yrityksen sisäisen Tekla- kehitysryhmän kautta.

Avainsanat

Tekla Structures, tietomalli, Visual Studio, C#, käyttöliittymä

Degree Programme in Civil Engineering
Bachelor of Engineer

Author	Roosa Saari	Year	2020
Supervisor	Heikki Ala-Louko		
Commissioned by	Optiplan Oy		
Subject of thesis	Prefix-checking tool for Tekla Structures		
Number of pages	45 + 7		

This thesis concentrated on the opportunities of developing building information modelling. The goal of this thesis was to develop a new prefix-checking tool to Tekla Structures for Optiplan. This checking-tool makes structural engineering easier and faster.

This study and developing process started with creating different models to Tekla Structures of columns, beams, plates, and walls. The preliminary code received from the company was tested with created models. The kind of cases which the program could not process were developed further. The program was developed and the user interface was edited in the Visual Studio with C# language. The program was tested in Tekla during the development process. The user interface was made user-friendly by placing the buttons reasonably and adding instructions for use.

The kind of cases which the program cannot process were for example the hollow-core slab and the steel parts with a subassembly prefix. The preliminary code was changed and the problem with the hollow-core slab was solved. The problem with the subassembly prefix was not solved perfectly. The finished program is added to Tekla Structures when the last problem is completely solved.

Key words

Tekla Structures, building information modelling, Visual Studio, C#, user interface

SISÄLLYS

1 JOHDANTO	8
2 KEHITYSTYÖN KUVAUS.....	9
2.1 Tarpeellisuus	9
2.2 Prosessin eteneminen	9
2.3 Haasteet	9
2.4 Ratkaisumahdollisuudet.....	10
3 TYÖVÄLINEET	11
3.1 Työvälineet	11
3.2 .NET Framework.....	11
3.3 Ohjelmointikieli C#	12
3.4 Microsoft Visual Studio	12
3.5 Tekla Structures.....	13
3.6 Tekla Open API	13
4 TIETOMALLINTAMINEN	14
4.1 Tietomallintaminen ja sen edut	14
4.2 Tietomallivaatimukset	15
4.3 Numerointisuositukset.....	16
5 OHJELMAN TESTAUS.....	19
5.1 Lähtökohdat	19
5.2 Työkalun testaus.....	20
6 TULOKSET.....	22
7 TYÖKALUN KEHITTÄMISPROSESSI.....	24
7.1 Ohjelmakoodin nykyinen rakenne	24
7.2 Valintanäppäinten toiminta.....	27
7.3 Kehittämisprosessi.....	29
8 KÄYTTÖLIITTYMÄ	33
9 OHJELMAN KÄYTTÖ	36
9.1 Ohjelman käyttö vaiheittain.....	36
9.2 Huomioita.....	38

9.3 Vuokaavio ohjelman käytöstä	39
10TYÖKALUN KÄYTTÖÖNOTTO.....	40
11POHDINTA	41
LÄHTEET.....	42
LIITTEET	45

ALKUSANAT

Haluan kiittää Optiplania mielenkiintoisesta opinnäytetyön aiheesta, joka oli haastava mutta opettavainen. Erityiskiitokset haluan välittää yrityksen puolelta työtä ohjanneelle kärsivälliselle, erittäin ammattitaitoiselle Timo Hervalle.

KÄYTETYT MERKIT JA LYHENTEET

Tekla	Tekla Structures
VS	Visual Studio

1 JOHDANTO

Opinnäytetyön tarkoituksena on perehtyä Tekla Structuresissa kohdattuihin tietomallintamisen ongelmakohtiin ja kehittämismahdollisuuksiin. Opinnäytetyössä syvennytään rakennusosien tunnuksiin liittyviin pulmiin ja mallinnustapoihin. Tunnuksista käytetään nimitystä prefix. Lähtökohtana opinnäytetyön tekemiselle toimii toimeksiantajalta saatu tehtävä.

Opinnäytetyön toimeksiantaja on Optiplan Oy, joka toimii valtakunnallisesti asunto-, toimitila- ja korjausrakentamisen suunnittelijana ja asiantuntijana (Optiplan 2017). Pääasiallinen tarve prefixien tarkistusohjelman luomiselle on se, että osa elementtimallinnustyökaluista luovat osia ilman etuliitteitä, jolloin mm. elementtien numerointi häiriintyy.

Kehitystyön tavoitteena on kehittää ja ohjelmoida yrityksen sisälle työkalu, jonka avulla mallinnusta voidaan nopeuttaa sekä eliminoida mahdolliset suunnitteluvirheet. Työkalun avulla elementtisuunnittelija paikallistaa ilman etuliitettä luodut elementit ja voi ohjelman avulla korjata ongelmakohdat. Tavoitteena on luoda toimintavarma prefixien tarkistusohjelma sekä tehdä käyttöliittymästä käyttökelpoinen.

2 KEHITYSTYÖN KUVAUS

2.1 Tarpeellisuus

Tietomallipohjainen suunnittelu on tuonut uusia ulottuvuuksia myös rakenne- ja elementtisuunnitteluun. Tulevaisuudessa tietomallintaminen tulee olemaan todennäköisesti pääasiallinen suunnittelumuoto. Suunnitteluohjelma Tekla Structures tarjoaa ohjelmointirajapinnan Tekla Open API:n, joka mahdollistaa mallinnusta avustavien ohjelmien ja komponenttien kehittämisen.

Erilaiset mallinnusta avustavat ohjelmat nopeuttavat suunnittelua sekä helpottavat käyttäjän työskentelyä. Prefixien tarkistustyökalun avulla ilman etuliitettä olevat elementit huomataan ja numerointiin liittyviä virheitä ei pääse tapahtumaan. Mikäli näitä ei huomata ajoissa, aiheuttaa se lisätyötä jälkikäteen.

2.2 Prosessin eteneminen

Tekla Structuresiin mallinnetaan mahdollisimman monipuolisesti erilaisia mallinnustilanteita. Tämän jälkeen toimeksiantajalta saadun logiikaltaan toimivan ohjelmakoodin avulla tehdään soveltuvuusselvitys. Selvityksessä koodin käytettävyyttä ja toimivuutta testataan Teklassa luotuihin malleihin. Erilaiset mallinnustapakukset ja komponenttien testaukset listataan taulukkoon sekä lisätään tulos ohjelman toimivuudesta. Ohjelman testauksen jälkeen sellaisille mallinnustapauksille, joita ohjelma ei tunnista tai osaa käsitellä, pyritään jatkokehittämään toteutamiskelpoiseksi. Kehitystyö tapahtuu Visual Studiossa C# -ohjelmointikielellä.

2.3 Haasteet

Kehitystyöprosessin aikana voidaan törmätä muutamiin haasteisiin. Kaikkia mallinnustapauksia ei välttämättä huomata mallintaa, jolloin myöhemmin sellaisiin mallinnustapauksiin kohdatessa ohjelma ei välttämättä toimi oikein. Lisähaastetta opinnäytetyön tekemiseen tuo ohjelmoinnin tuottamat haasteet, sillä aikaisempaa kokemusta ohjelmoinnista ei ole.

2.4 Ratkaisumahdollisuudet

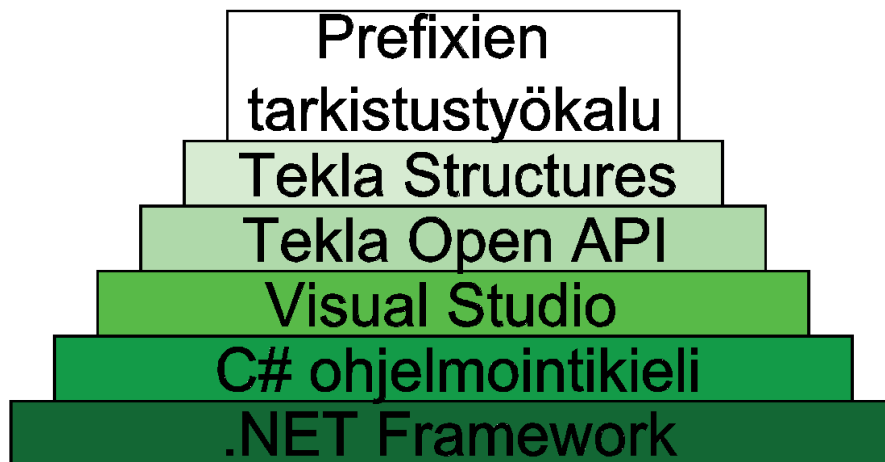
Työkalun testauksen yhteydessä voi tapahtua virheitä, jolloin testaustuloksesta voi tulla virheellinen. Tällaiset virheet eliminoidaan tekemällä testaus tarkasti ja tarvittaessa useaan otteeseen.

Ratkaisuja ohjelmoinnin tuottamiin haasteisiin voidaan hakea Ghodrat Moghaddampourin kirjoittamasta C# ohjelmointi -kirjasta, joka käsittelee perustapaukset C# -ohjelmoinnista. Monimutkaisiin ohjelmarakenteisiin apua on pyydettävä toimeksiantajalta. Mikäli kaikkiin mallinnustapauksiin ei löydetä ratkaisua, ohjelma jää vaillinaiseksi ja kaipaa myöhemmin jatkokehittämistä.

3 TYÖVÄLINEET

3.1 Työvälineet

Kehitysprosessi koostuu monesta osa-alueesta ja siihen tarvitaan erilaisia työvälineitä. Kuviossa 1 on havainnollistettu, kuinka ohjelman kehitykseen tarvittavat eri työvälineet linkittyvät toisiinsa. Kaiken pohjana on .NET sovelluskehys (.NET Framework), jonka päälle ohjelma rakentuu. Pyramidin huipulla on lopullinen prefixien tarkistustyökalu, joka tarvitsee oikein toimiakseen alla olevia alustoja.



Kuvio 1. Ohjelman rungon alustat

3.2 .NET Framework

Alun perin Windows-käyttöjärjestelmän kehitysympäristöksi luotu sovelluskehys .NET on suunniteltu toimivan ympäristöissä, joissa laitteet ovat internetin kautta välityksessä toisiinsa. Näin ollen laitteiston järjestelmätason mukaan ohjelmointirajapinta on yhteinen ja kaikkien saatavissa. Sovelluskehyksellä tarkoitetaan ohjelmistoa, joka toimii runkona sen päälle rakennettaville tietokoneohjelmille. .NET tarjoaa laajan toiminnallisuuksia sisältävän kirjaston, mikä nopeuttaa ohjelmoinnin kehitystyötä, kun valmiiksi rakennettuja osia ei tarvitse kirjoittaa uudelleen ohjelman kehityksen aikana. Erilaisten Windows-, web- ja mobiilisovellusten kirjoittaminen on .NET- ympäristössä mahdollista, joten se sopii hyvin myös prefixien tarkistustyökalun kehitysalustaksi. (Moghadampour 2012, 12–13.)

.NET- sovelluskehys rakentuu muutamista pääkomponenteista: ohjelmointikielistä ja työkaluista, ajonaikaisesta ympäristöstä sekä perusluokkakirjastosta. Työkaluna toimii Visual Studio 2019 ja sen ohjelmointikielenä on C#. Ajonaikainen ympäristö on turvallinen tiukkoine turvallisuus- ja tarkkaavaisuusmääräyksineen, sekä monipuolinen tarjotessaan erilaisia palveluita käyttäjälle. Perusluokka puolestaan sisältää laajan kokoelman valmiiksi kirjoitettuja luokkia, joita ohjelman kehittäjä voi käyttää hyväkseen. (Moghadampour 2012, 12–13.)

3.3 Ohjelmointikieli C#

Ohjelman kehittämisessä käytetään ohjelmointikieltä C#. Visual C# perustuu C++ kieleen, ja yhdistää C++:n laskentatehokkuuden ja Visual Basicin helppokäyttöisyyden (Moghadampour 2012, 14). Oliopohjainen C# käyttää tiedonsiirtovälineenään XML:ää (Extensible Markup Language). XML-kielen avulla voidaan luoda tietoja ja merkitä ne XML-tunnisteilla yhdessä järjestelmässä, jonka jälkeen tietoja voidaan käsitellä toisessa järjestelmässä käyttöjärjestelmästä ja laiteympäristöstä riippumatta (Microsoft Office 2020). Olioiden väliseen kommunikointiin C# käyttää SOAP:ia (Simple Object Access Protocol) (Moghadampour 2012, 14). SOAP tarjoaa tavan kommunikoida sovellusten välillä riippumatta käyttöjärjestelmästä, tekniikasta ja ohjelmointikielestä (w3schools 2020).

3.4 Microsoft Visual Studio

Kehitystyössä hyödynnetään Microsoftin tarjoamaa englanninkielistä ohjelmointikehitysympäristöä Visual Studio 2019. VS on Microsoftin RAD (Rapid Application Development) integroitu kehitysympäristö, jolla voidaan luoda monipuolisesti erilaisia .NET- sovelluksia (Moghadampour 2012, 15). Kehitysympäristössä voidaan kehittää muun muassa Windows, mobiili ja web- sovelluksia. Visual Studio tukee useita eri ohjelmakieliä, mukaan lukien C#. Visual Studiossa sovellusten kehittämistä helpottaa VS:n tarjoamat perusluokkakirjastot, sekä ajonaikainen virheiden korjaaja, joka etsii ja korjaa virheet riveittäin ja listaa havaitut virheet listaksi. (Visual Studio 2020.)

3.5 Tekla Structures

Tekla Structures on Trimblen kehittämä edistysellinen tietomalliohjelmisto rakennesuunnittelun työnkulkuun. Tekla Structuresilla voidaan mallintaa tarkasti koko hankkeen elinkaaren ajan erilaisia rakenteita. Teklalla luotuja tarkkoja, yksityiskohtaisia tietoja sisältäviä tietomalleja voidaan jakaa vaivattomasti ja työstää yhtä aikaa organisaation sisällä. Tekla sopii yhteen useiden ratkaisutoimittajien ohjelmistojen kanssa ja mallit on mahdollista linkittää IFC:n kautta arkkitehtuuri-, talotekniikka- ja tehdassuunnitteluohjelmistoihin. IFC (Industry Foundation Classes) tarkoittaa kansainvälistä rakennusalan standardin ISO 16739 mukaista tiedonsiirtoa tietokonejärjestelmästä toiseen (buildingSMART 2020). Tekla on saatavilla 17 kielellä 32 ympäristössä, jotka sisältävät etukäteen määritetyt aluekohtaiset asetukset ja tiedot (Tekla 2020a).

3.6 Tekla Open API

Tekla Open API, joka tulee sanoista Application Programming Interface, tarkoittaa ohjelmointirajapintaa. Ohjelmointirajapinnalla voidaan kehittää uusia sovelluksia (applications) ja laajennuksia (plug-ins) Tekla Structuresiin. Sovellusten ja laajennusten ero on niiden käyttämisessä. Laajennukset ovat komponenttityökaluja, joita käytetään mallintamisessa ja piirustuksissa (Trimble 2020b). Laajennukset käynnistetään Teklan sisällä komponenttikatalogista tai työkaluvalikosta, riippuen siitä onko kyseessä mallissa vai piirustuksessa käytettävä laajennus (Kuusela 2014, 23). Sovellukset puolestaan käynnistetään Teklan ulkopuolella, ellei niitä ole sisällytetty makroon. Tekla Open API tarjoaa siis rungon Teklaan liitettäville ohjelmille, jossa kehitettävän sovelluksen ohjelmakoodi kirjoitetaan Teklan ulkopuolelle ja linkitetään Visual Studion avulla Teklaan. Tekla Open API käyttää Microsoft.NET- tekniikkaa, jonka avulla Tekla Structures voidaan integroida muiden ohjelmistojen kanssa. (Tekla 2020c.)

4 TIETOMALLINTAMINEN

4.1 Tietomallintaminen ja sen edut

Rakennuksen tietomallilla (engl. Building Information Model, BIM) tarkoitetaan digitaalista rakennuksen virtuaalimallia, joka vastaa todellisuutta. Digitaaliset mallit pitävät sisällään tietoja, joita tarvitaan rakentamisen eri vaiheissa kuten osien valmistuksessa ja hankinnassa. Tietomalli ei ole siis pelkkää 3D- suunnittelua, vaan sisältää tietoja rakennuksesta koko rakennusprojektin elinkaaren ajalta. (Tekla 2020b.)

Yleisten tietomallivaatimusten Yleinen osuus Osa 1. on kuvannut tietomallinnuksen pää tavoitteita seuraavasti: *”Kiinteistöjen ja rakennuksien mallinnuksen tavoite on suunnittelun ja rakentamisen laadun, tehokkuuden, turvallisuuden ja kestävän kehityksen mukaisen hanke- ja elinkaari prosessin tukeminen”* (RT 2012. RT 10-11070, 2).

Todenmukaisen mallin ja täsmällisen geometrian avulla suunnitelmat ovat havainnollistavia ja helpommin ymmärrettävissä (Tekla 2020b). Mallia tarkastelemalla päästään laadukkaampaan lopputulokseen, kun virheet havaitaan ajoissa. Määrälaskennan avulla mallista saadaan kustannuslaskentaan tuoreet tiedot ja kustannusarvio on helpommin arvioitavissa (RT 2020. RT 10-11070, 4).

Tietomalli koostuu useista eri suunnittelualojen malleista, joten törmäystarkaste-luita voidaan tehdä eri suunnittelualojen mallien välillä helposti (Elementtisuunnittelu 2020). Perinteiseen 2D- suunnitteluun verrattuna muutosten hallinta on huomattavasti helpompaa ja nopeampaa. Siinä missä kaksiulotteisessa suunnittelussa tiedon päivittäminen täytyy tehdä erikseen jokaiseen piirustukseen, laskelmiin ja detaljeihin, tietomallipohjaisessa suunnittelussa päivittäminen tapahtuu yhdestä mallista.

Riskianalyysin parantumisen sekä aikataulu- ja kustannuslaskennan helpottumisesta huolimatta tärkein ominaisuus avoimessa tietomallintamisessa on yhteistyön lisääntyminen eri osapuolten välillä (Kuvio 2). Malli on reaaliajassa kaikkien hankkeen eri osapuolten käytettävissä, jolloin muutokset näkyvät suoraan myös

työmaalla (Tekla 2020b). Mallia voidaan käyttää koko elinkaaren ajan, ja on hyödyksi esimerkiksi rakennuksen ylläpitovaiheessa vuorovaikutteisen huoltokirjan muodossa (RT 2012. RT 10-11070, 4).



Kuvio 2. Avoin tietomallintaminen (Tekla 2020b)

4.2 Tietomallivaatimukset

Rakennustieto on julkaissut ohjeet kansallisiksi tietomallivaatimuksiksi. Jotta mallinnus onnistuu odotetusti, malleille on asetettava hankekohtaiset painopistealueet ja tavoitteet. Tavoitteiden ja yleisten tietomallivaatimusten julkaisusarjan pohjalta määritetään ja dokumentoidaan projektikohtaiset vaatimukset. Mallien tarkkuus ja sisältövaatimukset muuttuvat suunnitteluvaiheen mukaan. Yleisten tietomallivaatimusten osa 5:n täydentävä liite käsittelee yksityiskohtaisemmin mallinnustapaa ja tarkkuustasoa (Taulukko 1). (RT 2012. RT 10-11080, 2).

Yleisiä vaatimuksia mallinnettaville rakenteille on, että rakennemallin täytyy sisältää kaikki kantavat ja ei-kantavat betonirakenteet. Lisäksi rakenteet tulisi mallintaa siten, että tietoa siirrettäessä rakennusosan sijainti, tyyppi ja geometria siirtyvät rakennusosan mukana. Myös kaikki sellaiset tilaa vievät rakennustuotteet, joiden koolla ja sijainnilla on merkitystä muille suunnittelijoille, täytyy mallintaa. Näin ollen talotekniikkasuunnittelijoiden työ helpottuu, kun rakennustuotteiden oikeat korot ovat selvillä. (RT 2012. RT 10-11080, 2).

Taulukko 1. Ote mallinnuksen tarkkuustasosta (RT 2016. RT 10-11209, 4)

Mallinnettavat rakennusosat sekä mallinnuksen tarkkuustaso				IFC-mallin kappaleiden tietosisältö																							
Nro	Osat	Hankintoja palveleva suunnittelu	Kommenttikenttä	Nimi	Profiili	Kerros	Materiaali	Tunnus	Status-tieto	Lohko	Juokseva nro (ACN)	Precast/Cast in place	Luokittelu (class)	Ylin korkeusasema	Alin korkeusasema	Ylin globaali kork.as.	Alin globaali kork.as.	Pinta-ala (brutto)	Pinta-ala (netto)	Pituus	Leveys	Korkeus/paksuus	Tilavuus	Betoniosien tilavuus	Paino [kg]		
122	Alapohjat	3	Kaadoilla	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1221	Maata vasten valettu kantava laatta	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1221	Elementtilaatat	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1221	Maanvarainen laatta	2	Kaadoilla	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		
	Lämmöneristeet	2		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x				
1222	Alapohjakanaalit	2		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1223	Erityiset alapohjat (esim. luiskat ja uimaallasrakent.)	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
123	Runko																										
1231	Väestönsuojat	3		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1232	Kantavat seinät	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1233	Pilarit	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1234	Palkit	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1235	Väliopohjat, elementit	3		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1235	Väliopohjat, paikallavalurakenteet	3	Paikallavalukentät jaetaan tuotannon määrittelemiin valualueisiin	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1236	Yläopohjat, kantava rakenne	3	Mallinnetaan kuten väliopohjat, kattoluukkujen varaukset mallinnetaan	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
	Yläopohjat, ristikot	3		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		
	Äänen-, lämmön- ja paloeristykset	2	Mallinnetaan kaikki tilaa vievät rakennustuotteet, joiden koolla ja sijainnilla on merkitystä muille suunnittelijoille	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x				
1237	Runkoportaat, paikallavaletut	3		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x		

Yleisten tietomallivaatimuksien lisäksi betonielementtiteollisuus, rakennesuunnittelijat ja Tekla Oyj kehittivät yhteistyössä BEC 2012 projektissa elementtisuunnittelun mallinnusohjeen. Ohjeistuksen tarkoitus on ohjeistaa betonielementtien tietomallinnusta, joita kaikkien elementtejä mallintavien tulisi noudattaa. Tällöin mallit ovat samanlaisia suunnittelutoimistosta tai mallintajasta riippumatta. (Betoniteollisuus ry 2012, 4.)

4.3 Numerointisuositukset

Teklan suomiympäristön luonnin yhteydessä Teklan ja kaikkien asiakkaiden yhteisenä projektina on sovittu yhteiset numerointisuositukset mallintamiselle (Nieminen 2012). Numerointisuositus kattaa ohjeistuksen teräsosien lisäksi betonielementtien osille, erilaisille paikallavaluosille sekä raudotteille. Useimmilla, erityisesti suuremmilla mallinnusta tarjoavilla suunnittelutoimistoilla on suomiympäristön numerointisuositukseen pohjautuvia omia muokattuja ohjeistuksia.

Taulukosta 2 nähdään, kuinka numerointisuosituksessa on listattu suositeltu nimeäminen suomeksi ja englanniksi, class- arvo sekä prefix- tiedot. Teklan valmiit

valinta- ja näkymäsuodattimet, kuten oletusosien profiilit, perustuvat Teklan suomiympäristön numerointisuositukseen (Nieminen 2012).

Taulukko 2. Ote numerointisuosituksesta (Tekla 2020)

Nimi	Name	Class	Cast Unit	Part	Plate	Rakenneosa	Structure	BETONELEMENTTI / CONCRETE ELEMENT
Runkoelementit	Framework elements	201-209	Prefix and start number					Oletusosa / default profile
PILARI	COLUMN	201	P1			Suorakaidepilari	Rectangle beam	380*380
PILARI	COLUMN	202	P1			Pyöreä pilari	Round column	D380
SUORAKAIDEPALKKI	RECTANGLE_BEAM	203	JK1			Jännebetonipalkki, suorakaide	Rectangle beam, prestressed	780*380
SUORAKAIDEPALKKI	RECTANGLE_BEAM	204	K1			Suorakaidepalkki	Rectangle beam	780*380
LEUKAPALKKI	GNATHIC_BEAM	205	JK1			Jännitetty leukapalkki	Gnathic beam, prestressed	RCL300*600-400*150
LEUKAPALKKI	GNATHIC_BEAM	206	K1			Leukapalkki	Gnathic beam	RCL300*600-400*150
MATALALEUKAPALKKI	LOW_GNATHIC_BEAM	207	JK1			Matalaleukapalkki	Gnathic beam, low height	RCDL280*375*580*100*100
HI-PALKKI	RIDGE_I-BEAM	208	HI1			HI-palkki	Ridge I-beam	
I-PALKKI	I-BEAM	209	I1			I-palkki	I-beam	

Täydentävä listaus elementtitunnuksiin liittyen löytyy elementtisuunnittelun sivuilta elementtitunnukset 2011 (Taulukko 3), johon on listattu erilaiset elementit ja annettu suositellut elementtitunnukset (Elementtisuunnittelu 2020).

Taulukko 3. Ote elementtitunnuksista (Elementtisuunnittelu 2020)

ELEMENTTITYYPPI	ELEMENTTI	TUNNUS
PERUSTUSELEMENTIT	ANTURAELEMENTTI	A
	PILARIHOLKKIELEMENTTI	PH
	SOKKELIELEMENTTI (EI KANTAVA)	AN
	SOKKELIELEMENTTI (KANTAVA)	AS
	SOKKELIPALKKI	AK
	SOKKELIRUUTUELEMENTTI (MAANPAINE)	AR
	SOKKELIELEMENTTI (MAANPAINE, YKSI KUORI)	AV
	TUKIMUURIELEMENTTI	TKE

Teklassa elementtinumeroinnin seuraaminen voi olla haastavaa, kun samanlaiset elementit saavat saman numerotunnisteen. Elementtiteollisuus suosittelee käytettäväksi erillistä id-numerointia, jolla yksilöidään yksittäinen elementti (Nieminen 2012). Elementtejä voidaankin seurata elementtitunnusten sijaan ACN-numeroinnilla, (Assembly Control Number) joka tarkoittaa juoksevaa yksilöllistä numerointia. ACN-numerointi säilyy elementillä läpi projektin, vaikka elementtitunnuksen ja/tai piirustusnumeron numerotunniste suunnittelun aikana vaihtuisikin. Yksilöllinen ACN numero esitetään piirustuksissa elementtitunnuksen lisäksi. Tämä helpottaa tuotannon suunnittelua ja ohjausta, sekä elementtien asennusta. (Betoniteollisuus ry 2012, 14.)

Elementtitunnusten numerointiin liittyen suunnittelutoimistojen välillä voi olla eroavaisuuksia. Osa urakoitsijoista, elementtitehtaista ja -suunnittelijoista suosivat mieluummin yhtä elementtitunnusta esim. AN-4001, missä ACN on numero-

osa. Tällöin jokainen elementti saa oman elementtipiirustuksensa, vaikka olisivat keskenään samanlaisia. Yhdessä piirustuksessa voidaan sujuvasti esittää samanlaiset elementit, kun elementtinumeron lisäksi on esitetty erillinen ACN-numero.

5 OHJELMAN TESTAUS

5.1 Lähtökohdat





Lähtökohtana ohjelman testaukselle on kokeilla toimeksiantajan ohjaajalta saatua logiikaltaan toimivaa ohjelmaa Teklassa luotuihin erilaisiin rakennemalleihin. Tekla-malliin luodaan erilaisia tapauksia pilareista, palkeista, laatoista ja seinistä. Mallinnusta pyritään tarkastelemaan mahdollisimman monesta eri näkökulmasta, monella eri työkaluilla ja tavoilla. Testauksen jälkeen saadaan tuloksia, kuinka logiikaltaan toimiva prefixien tarkistustyökalu toimii tällä hetkellä.

Suuressa roolissa testauksen suhteen ovat Teklan sisältämät erilaiset komponentit, kuinka ne käyttäytyvät ehjinä ja räjäytettyinä. Komponentit ovat erääläisiä valmiita paketteja, joihin on sidottu erinäisiä toiminnallisuuksia; esimerkkinä pilari-palkki-liitäntä, jossa mallin eri osat liittyvät yhteen komponentin avulla. Kun komponentti räjäytetään, siitä poistuu komponentin parametrit eli ”äly”, ja jäljelle jää Teklan perusosien toiminnallisuus. Erilaisten tilanteiden testauksesta saadaan tuloksia, missä tilanteessa ohjelma toimii ja missä ei.

Tekla Structuresilla mallintaessa jokaiselle osalle annetaan prefix. Prefixeillä tarkoitetaan tunnusta, jolla kerrotaan minkä tyyppinen rakenneosa on kyseessä, kuten pilari, palkki tai sandwich- seinäelementti. Prefix- tunnukset annetaan numerointisuosituksen listauksen mukaisesti (Taulukko 2). Betoniosien prefixit pohjautuvat elementtisuunnittelun mallinnusohjeeseen (Taulukko 3). Tekla numeroi kokoonpanot ja osat annettujen tunnuksien ja mallinnustavan mukaisesti. Kokoonpanotasolla prefix määräytyy pääosan mukaan.

Tutkimustyössä hyödynnetään eri valintatasoja, joita Teklassa on neljä. Eri valintatasojen käyttö helpottaa testausprosessia. Näistä kaksi on komponenteille ja kaksi kokoonpanotasolle. Kokoonpanolla (assembly) tarkoitetaan rakennetta, joka koostuu yhdestä tai useammasta osasta tai muusta tarvikkeesta, kuten pilarikokoonpanossa olevat pilarikengät ja pilarikonsoli. Taulukkoon 4 on koottu Teklasta löydettävät valintatasot symboleineen ja selityksineen.

Taulukko 4. Tekla Structures valintatasot

	Select components	Komponenttien valintanäppäin, joka valitsee koko komponentin, siitä huolimatta mitä komponentin objektia napsautetaan.
	Select objects in components	Tällä valinnalla voidaan valita objekteja komponenteista. Esim. ulkokuori sandwich-elementistä.
	Select assemblies	Kokoonpanovalinta. Mitä tahansa kokoonpano- osaa napsauttamalla kaikki kokoonpanoon kuuluvat objektit aktivoituvat.
	Select objects in assemblies	Valinnalla voidaan valita yksittäisiä objekteja kokoonpanosta. Esim. pilarikenkien valinta pilarikokoonpanossa.

5.2 Työkalun testaus

Käytännössä työkalun testaus aloitetaan luomalla taulukkoon tilanteet ja tavat, joissa työkalun halutaan toimivan. Pääosalta puuttuu harvoin prefix, joten testauksen pääpaino on komponenteissa ja moniosaisissa rakenteissa. Listaus on tehty toimeksiantajan tehtävälistan mukaan. Jokaisen mallinnetun osan toimivuutta kokeillaan, ja kirjataan ylös tulokset, kuinka osat käyttäytyvät työkalua kohteilla (Taulukko 5). Testaus on pyritty tekemään selkeässä järjestyksessä yksi osa-alue kerrallaan pilarikonsoleilta, laatoilta ja seiniltä. Virheet on pyritty eliminomaan tekemällä testaus useaan kertaan, mutta siitä huolimatta ne ovat mahdollisia. Taulukon alapuolelle on koottu selitykset taulukoissa käytetyille merkeille.

Taulukko 5. Työkalun testaus.

TYÖKALUN TESTAUS	Komp. Ko- koonp.	Komp. ei ko- koonp.	Räj.komp. kokoont.	Räj.komp.ei kokoont.
Pilarikonsolit ja pilarit				
Pilarikonsoli pilarin päässä (reunapilari)	X	*	*	*
Pilarikonsoli pilarin keskellä (reunapilari)	X	*	*	*
Pilarikonsoli pilarin keskellä (keskipilari)	X	*	*	*
Laatat				
Massiivilaatan lisätty materiaali	X	*	X	*
Ontelolaatta kylpyhuoneen syvennyksen vahvikepalkki	-	-	-	-
Seinät				
Sandwich-seinä ikkuna-aukko reunavahvistukset	X	X	X	X
Pilaster- paksunnos sisäkuoressa	X	*	*	*
Sisäkuorielementin kaideholkki-putki	-	-	-	-
Valutarvikkeet				
SBKL- vakiokiinnityslevy	-	-	-	-
WELDA Strong- kiinnityslevy	-	-	-	-
WELDA Strong- kiinnityslevy	-	-	-	-

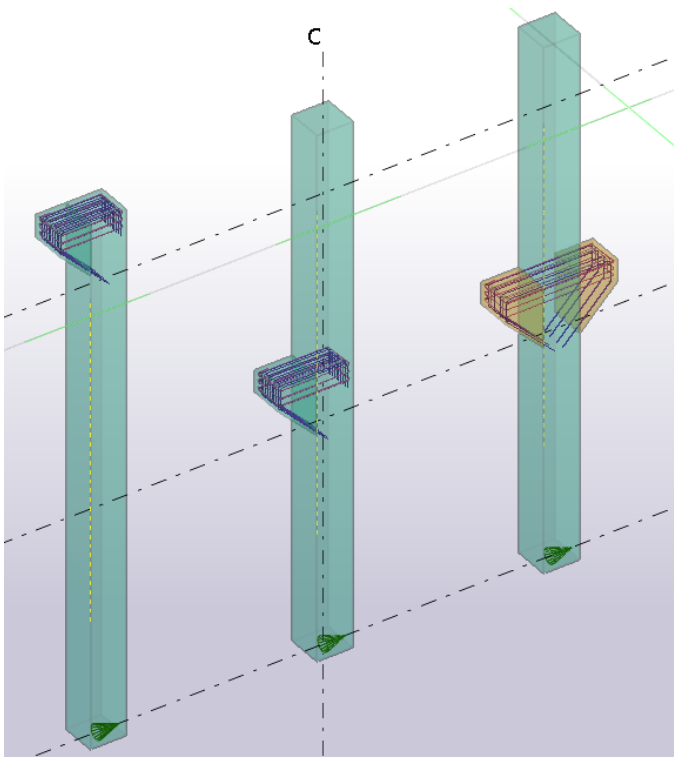
x= Ohjelma toimii odotetusti.

*= Pääosalta puuttuu prefix.

-=Ohjelma ei toimi oikein.

6 TULOKSET

Pilarikonsoleita tehdään kolme erilaista tapausta: pilarikonsoli pilarin päähän, yksipuolinen pilarikonsoli pilarin keskellä sekä molemminpuolinen pilarikonsoli pilarin keskellä (Kuvio 3). Testauksessa huomataan, että ohjelma toimii oikein, kun konsoli kuuluu räjäyttämättömänä komponenttina samaan kokoonpanoon pääosan eli pilarin kanssa. Räjäytettynä sekä kokoonpanosta irrotettuna ohjelma ei toimi oikein ja antaa virheilmoituksen: "Pääosalta puuttuu prefix!". Näin tapahtuu, koska konsoli on irrotettuna pilarista ja ohjelma olettaa konsolin olevan pääosa.



Kuvio 3. Pilarikonsolit

Massiivilaatalle lisätyn materiaalin kanssa ohjelma käyttäytyy oikein, kun ne ovat samassa kokoonpanossa, mutta erillisenä assemblyna ohjelma tulkitsee lisätyn materiaalin pääosana.

Pilaster-paksunnokselle käy samanlailla kuin pilarikonsoleille, ohjelma tulkitsee paksunnoksen yksittäiseksi pääosaksi ja antaa virheilmoituksen pääosan prefixin puuttumisesta. Kaideholkkiputken kohdalla huomataan, että ohjelma ei toimi sellaisille komponenteille, joilla on kaksi eri prefixiä, niin sanottu alikomponentin prefix (Kuvio 4). Ohjelma ei reagoi, vaikka osanumerointi puuttuu. Ohjelma kuitenkin

korostaa osan, jos kokoonpanon numerointi puuttuu, mutta ei korjaa prefixiä vaan jättää osan mustaksi. Näin ollen tällaiset tapaukset kaipaavat vielä lisäkehitystä.



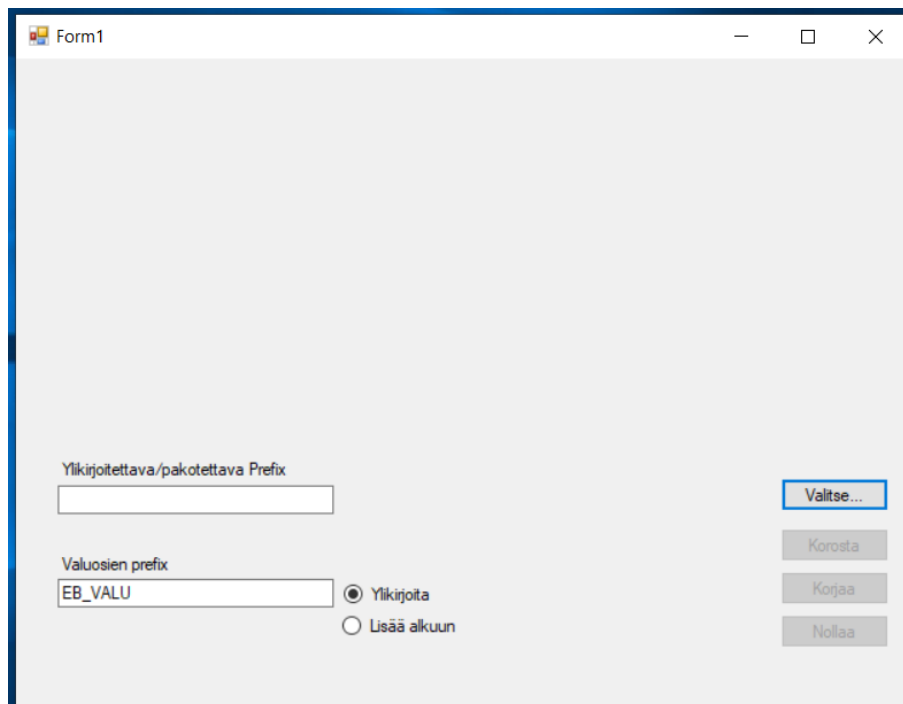
Kuvio 4. Esimerkkitapaus alikokoonpano prefixistä

Elementtien lisäksi tehdään testausta myös erilaisille valuosille, jossa käytetään valutarvikkeita, kuten erilaiset valulevyt. Huomataan että ohjelma ei toimi myöskään valuosien tapauksissa oikein kahden eri prefixin vuoksi.

7 TYÖKALUN KEHITTÄMISPROSESSI

7.1 Ohjelmakoodin nykyinen rakenne

Nykyisen ohjelman logiikka perustuu siihen, että ohjelma yrittää automaattisesti korjata prefixejä "Korjaa"- napin painalluksesta. Kuviossa 5 on esitetty alkuperäinen ohjelma yksinkertaisuudessaan, johon on lisätty tekstikentät pakotettavan prefixin ja valuosien prefixien osalta, joilla prefixiä voidaan muuttaa käsin.



Kuvio 5. Alkuperäinen ohjelma

Seuraavassa käydään läpi ohjelmakoodin rakennetta ja toiminnallisia osia aihealueittain yleisesti.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Tekla.Structures;
using Tekla.Structures.Model;
using Tekla.Structures.Model.UI;
```


Alussa määritellään tarvittavat *using*- rakenteet eli referenssit ohjelmaan. Referenssit ovat viittauksia .dll-tiedostoihin, joita voidaan hyödyntää ohjelmassa. Dynamic Link Library (.dll) -tiedostot ovat kirjastotiedostoja, jotka sisältävät kokoelman kutsuttavia funktioita ja objektityyppejä. (Reviversoft 2020).

```
namespace PartPrefixit
```

Nimiavaruus *namespace* on pakollinen osa C# -ohjelmaa. Nimiavaruuden avulla uuden esittelyalueen määrittelemine on mahdollista. Nimiavaruuksien avulla ohjelma voidaan tarvittaessa jakaa eri osioihin, joille annetaan osiota kuvaava nimi. Mikäli nimiavaruuden nimeä ei haluta käyttää luokan nimen kanssa, voidaan nimiavaruus ottaa käyttöön *using*- lauseella. (Moghadampour 2012, 190, 197.)

```
public partial class Form1 : Form
```

Luokka *class* kuvaa oliojoukkoa, jolle määritellään yhteiset attribuutit ja metodit (Moghadampour 2012, 134–135). Saantimääre *public* viittaa luokan jäseneen, joka on saatavilla rajoituksetta. ”Osittaista luokitusta” *partial class* voidaan pitää C#:n erityispiirteenä. Tällöin yhden luokan toiminnallisuutta voidaan toteuttaa useissa tiedostoissa ja kaikki nämä tiedostot yhdistetään yhdeksi luokkatiedostoksi sovellusta käännettäessä (Geeks for geeks, 2020). Luokkamäärittely on pakollinen osa ohjelmaa, sillä se mm. määrittää ohjelman tyyppin.

```
Model malli = new Model();
```

Luokan määrittelyn jälkeen olioita voidaan luoda *new*- operaattorilla. Olion luonnin jälkeen sillä on luokan mukainen tietorakenne, sekä luokassa esitetyt attribuutit ja metodit (Moghadampour 2012, 140). Malli-objekti sisältää kaikki malli-objektin käytettävissä olevat metodit. Tämä komentorivi linkittää Tekla-mallin ja suoritettavan .exe-ohjelman toisiinsa.

```
public Form1()
{
    InitializeComponent();
}
```

Funktio *Form1()* ja menetelmä *IntializeComponent()* alustavat ohjelman käyttöliittymän. Käyttöliittymän graafinen asettelu yhdistetään ohjelmakoodiin, ja toiminto aikaansaa käyttöliittymän näkymisen. Käyttöliittymän objektien asetukset, ja mm. painikkeiden lisäykset tapahtuvat tämän toiminnon kautta. Suunnittelija voi vaikuttaa käyttöliittymän visualisointiin tässä. (Dot Net Perls 2020.)

```
private string GetPartPrefix(ModelObject modelobject)
{
    switch (modelobject)
    {
        case Beam _:
            //Muunnetaan ModelObject Beamiksi

            //Luetaan osan Prefix ja palautetaan se

        case ContourPlate _:
            //Muunnetaan ModelObject Beamiksi ja Contourplateksi

            //Luetaan osan Prefix ja palautetaan se

        default:
            return "";
    }
}
```

Funktio palauttaa lähtöarvona annetun *ModelObjectin* prefixin, joka on perusluokka kaikille malliobjekteille. Merkkijono *string* on C# -kieleen valmiiksi kirjoitettu luokka, johon kuuluu myös paljon erilaisia sisäänrakennettuja metodeja tekstin käsittelemiseksi. Saantimääre *private* viittaa siihen, että pelkästään funktion sisältämät yksityiset muuttujan voivat käsitellä niitä. (Moghadampour 2012, 31, 135.)

Vertailulauseella *switch* voidaan tutkia yhden muuttujan tai lausekkeen arvon eri vaihtoehtoja. Sillä voidaan korvata peräkkäiset *if*- ja *else if* -lauseet. *Switch*-lauseeseen kuuluu joukko peräkkäisiä *case*- haaroja, joilla verrataan tutkittavan muuttujan arvoa ennalta määrättyihin arvoihin. Vertailulauseessa siirrytään seuraavaan *case*-haaraan hyppylauseen *return* avulla. Vertailutulosten perusteella suoritetaan eri toimenpiteitä, ja mikäli *case*-haarat epäonnistuvat, *default*-haara suoritetaan. (Moghadampour 2012, 105, 116.)

7.2 Valintanäppäinten toiminta

```
private void button1_Click(object sender, EventArgs e)
{
    ...
}
```

Object sender viittaa toimintoon, joka suoritetaan, kun ohjelman käyttäjä painaa painiketta `button1`, "Valitse..". *EventArgs* toimii perustana luokille, jotka sisältävät tapahtumadataa ja tarjoaa ominaisuuksia tarvittavien tietojen tallentamiseksi (Microsoft 2020). Nämä kaksi komentoa yhdessä tekevät halutun toiminnon.

```
ModelObjectEnumerator enumeraattori = picker.PickObjects(Picker.PickOb-
jectsEnum.PICK_N_PARTS, "Valitse osat. Kuittaa hiiren keskipainikkeella.");

while (enumeraattori.MoveNext())

{

//Tutkitaan nykyisen objektin prefix

}
```

Toiminto ohjataan Tekla-malliin ja seurataan alareunassa näkyvän dialogin ohjetta "Valitse osat. Kuittaa hiiren keskipainikkeella." Poiminta- luokan *picker* avulla kohteita voidaan valita manuaalisesti Tekla-mallista (Trimble 2020e). Pickerin määritteissä valinta rajoitetaan vain parteihin. Enumeraattori on eräänlainen luettelo, joka sisältää kaikki Teklan ikkunassa äskettäin valitut objektit. Se sisältää *MoveNext()*- ja *Current()*- toiminnot, jolloin malliobjekteja iteroidaan niin kauan, kuin objekteja on jäljellä (Trimble 2020f). Jos valittavia objekteja ei ole enää jäljellä, valinta saa arvon nolla. Tässä hyödynnetään toistolauseetta *while*. Sen avulla samoja lauseita toistetaan niin kauan kuin määrätty ehto on tosi tai numeerinen arvo poikkeaa nolasta (Moghadampour 2012, 109).

Mikäli käyttäjä klikkaa taustanäkymää, ohjelma antaa virheilmoituksen: "Valinta on tyhjä!" Tämä toiminto saadaan näkyviin *Message.show* -komennolla. Poikkeustilanteita, toisin sanoen virhetilanteita käsitellään *try-catch*-rakenteella.

```
try
{
    ...
}
catch (Exception)
{
    //Näytetään ilmoitus "Valinta on tyhjä!", jos valinta osuu taustanäkymään
}
```

Ohjelmassa pyritään ensin suorittamaan *try*-lohkossa olevat lauseet, ja virhetilanteen sattuessa siirrytään suorittamaan *catch*-lohkon lauseet. (Moghadampour 2012, 370.) *Catch*-rakenteessa voidaan esimerkiksi esittää virheilmoitus, ja ohjelman suorituksen pitäisi jatkua virheestä huolimatta. Ilman *try-catch*-rakennetta ohjelma kaatuisi virhetilanteeseen.

```
private void button2_Click(object sender, EventArgs e)
{
    ModelObjectVisualization.SetTemporaryState(puuttuvat, new Tekla.Structures.Model.UI.Color());
}
```

Funktio suoritetaan käyttäjän painaessa painiketta *button2*, ”Korosta”, jolloin ilman prefixiä olevat malliobjektit visualisoituvat väliaikaisesti mustaksi. *ModelObjectVisualization* viittaa visualisoinnin muuttumiseen ja *SetTemporaryState* väliaikaisuuteen. (Trimble 2020c).

```
private void button3_Click(object sender, EventArgs e)
{
    ...
    foreach (ModelObject muutettava in puuttuvat)
    {
        switch (muutettava)
        {
            case Beam _:
                //Jos muutettava on palkki
                }
                palkki.Modify();
                break;

            case ContourPlate _:
                //Jos muutettava on laatta
                laatta.Modify();
                break;

            ...
        }
        malli.CommitChanges();
    }
}
```

”Korjaa” -painikkeella *button3* ilman prefixiä olevat valitut objektit käydään vuorotellen läpi *foreach*-komennolla ja korjataan elementin pääosan mukaan *assembly.GetMainPart* avulla. Kaikki puutteelliset malliosat voidaan ajatella joko palkkeina *beam* tai laattoina *ContourPlate*. *Modify*-metodi on tärkeä, sillä se palauttaa mallitietokannan objektit vastaamaan olemassa olevaa objektia, ja tekee muutokset ohjelman virtuaaliseen malliin. *CommitChanges()*-toiminto suorittaa samat muutokset todelliseen malliin (Trimble 2020g).

```
private void button4_Click(object sender, EventArgs e)
{
    ...
    ModelObjectVisualization.ClearAllTemporaryStates();
}
```

Käyttäjän klikatessa painiketta button4 “Nollaa”, ohjelma poistaa väliaikaisen visualisoinnin kaikista malliobjekteista ja palauttaa pysyvän ulkoasun (Trimble 2020d). Tätä toimintoa voidaan hyödyntää, kun klikataan vahingossa väärää objektia ja halutaan päästä nopeasti alkutilanteeseen.

7.3 Kehittämisprosessi

Huomataan että alikokoonpano prefixien käsittely automaattisesti on vaikeaa. Ohjelman logiikkaa on muutettava niin, että käyttäjä itse korjaa kaikki prefixit selaamalla niitä ja korjaamalla ne yksi kerrallaan. Ohjelman alustus eli using-rakenteet, namespace sekä käyttöliittymän alustus säilyvät ennallaan.

```
private string GetPartPrefix(ModelObject modelobject)
{
    switch (modelobject)
    {
        case Beam _:
            //Muunnetaan ModelObject Beamiksi
            //Luetaan osan prefix ja palautetaan se
        case ContourPlate _:
            //Muunnetaan ModelObject Contourplateksi
            //Luetaan osan prefix ja palautetaan se
        case PolyBeam _:
            //Muunnetaan ModelObject PolyBeamiksi
            //Luetaan osan prefix ja palautetaan se
        case Brep _:
            //Muunnetaan ModelObject Brepiksi
            //Luetaan osan prefix ja palautetaan se
        default:
            return "";
    }
}
```

Tähän rakenteeseen on lisätty case-haarat Polybeam sekä Brep. Samat lisäykset on tehty myös *GetMainPart(ModelObject modelobject)* -rakenteeseen, joka valitsee ja palauttaa pääosan. Polybeamilla tarkoitetaan palkkia, joka koostuu monesta osasta. Brep puolestaan viittaa sanaan *boundary representation*, joka tarkoittaa pintojen välistä rajaa (RhinoCommon API 2020).

```
private void current()
{
    if (puuttuvat.Count != 0)
    {
```

```

        fiksattava.Clear();
        current_modelobject = puuttuvat[index];
        fiksattava.Add(current_modelobject);
        //Luetaan prefixit
        textBox3.Text = prefix;
        textBox4.Text = mainpartprefix;
        valitsin.Select(fiksattava);
        malli.CommitChanges();
    }
}

```

Tässä rakenteessa luetaan puuttuvien listan nykyinen mahdollinen objekti ja tehdään siitä fiksattava. Fiksattavien listan läpikäytävien objektien lukumäärä tulee näkyviin käyttöliittymän yläreunaan.

```

private void luetteloi()
{
    ModelObjectVisualization.SetTransparencyForAll(TemporaryTranspa-
rency.SEMITRANSSPARENT);
    //Asetetaan puuttellisten objektien korostusväri mustaksi
}

```

Komennolla *ModelObjectVisualization.SetTransparencyForAll(TemporaryTransparency.SEMITRANSSPARENT)* selattavan osien luetteloa selaamalla kyseinen objekti korostuu muusta mallista mustalla ja loput malliobjektit muuttuvat läpikuultaviksi. Toiminto helpottaa tilanteissa, joissa mallinnettuja osia on paljon ja ne sijaitsevat keskellä rakennusta muiden osien ympäröimänä.

Jotta ohjelman käyttö olisi toimintavarmempaa ja käyttäjäystävällisempää, määritetään tarkasteltavalle kohteelle zoomausalue. Toisin sanoen määritetään kohteelle eräänlainen pintamalli, josta poimitaan ääripisteet. Pisteiden avulla Tekla rajaa näkymän. Näkymärajauksessa on pyritty tekemään sopivan kokoinen huomioimalla sekä pienet valuosat että suuremmat betonirakenteet. Näkymärajauksessa hyödynnetään *GetSolid()*-komentoa.

Painikkeiden toimintaperiaate säilyy samana, vaikka sisältö hieman muuttuu. Toimintaperiaatteesta on kerrottu alaluvussa 7.2. Poimintatyökalulla (*picker*) valitaan Teklan ikkunasta puutteelliset objektit, jotka enumeroidaan eli luetteloidaan, ja jatketaan niin kauan, kuin objekteja on jäljellä. Enumeraattori on siis tässä tapauksessa jono *ModelObject*jeja. *ModelObject current = enumeraattori.Current()*; rivillä luodaan uusi *ModelObject*, jotta tarkasteltavaan *ModelObject*tiin voidaan viitata ja sen tietoja on mahdollista lukea. *ModelObject*ien jatkotoimenpiteet riippuvat sen classista, joten classeihin täytyy viitata lisäämällä varsinaisen classin perään välilyönti ja alaviiva.

```

private void button1_Click(object sender, EventArgs e)
{
    Picker picker = new Picker();
...
    try
    {
        ModelObjectEnumerator enumeraattori = picker.PickObjects(Picker.PickObjectsEnum.PICK_N_PARTS, "Valitse osat. Kuittaa hiiren keskipainikkeella.");
        while (enumeraattori.MoveNext())
        {
            ModelObject current = enumeraattori.Current;

            try
            {
                switch (current)
                {
                    case Beam _:
                        //Luetteloidaan valitut puutteelliset Beemit
                        ...
                }
            }
        }
    }
}

```

Alkuperäinen logiikaltaan toimiva ohjelma tuotti ongelmia juuri alikokoonpano prefixien osalta, kun prefixejä yritettiin korjata automaattisesti "Korjaa" -painikkeella. Nykyisen malliobjektin kokoonpanon alikokoonpanoille luodaan oma ArrayList, joka lisään kokoelmaan.

```

try
{
    //Hae aliassemblyt
    if (aliassemblyt.Count > 0)
    {
        foreach (Assembly alikokoonpano in aliassemblyt)
        {
            //Hae aliosat
            foreach (ModelObject sekundaari in sekundaarit)
            {
                if (!partlista.Contains(sekundaari))
                {
                    partlista.Add(sekundaari);
                }
            }
        }
    }
}
catch (Exception)
{
    throw;
}

```

Kaikki aliosat käydään läpi prefixien osalta; jos prefix on tyhjä, aliosa lisätään puutteellisten osien luetteloon. Foreach-loop käy seuraavassa läpi aliosaluettelon jokaisen ModelObjektin. Kun enumerointi ei ole mahdollista, foreach on erittäin

käyttökelpoinen tapa. Prefix-merkkijono alustetaan tyhjäksi ja nollataan mahdollinen aiemman osan prefix-luku.

```
foreach (ModelObject malliobjekti in partlista)
{
    prefix = "";
    ...
    try
    {
        //Luetaan osan prefix
        //Jos prefix on tyhjä, lisätään osa luetteloon
    }
    catch (Exception)
    {
        throw;
    }
}
```

"Edellinen" ja "Seuraava" -painikkeiden toiminnassa on hyödynnetty Index-muuttujaa. Alla on ote "Edellinen"- painikkeen toiminnasta. "Seuraava" -painike toimii päinvastoin index-muuttujan mukaan.

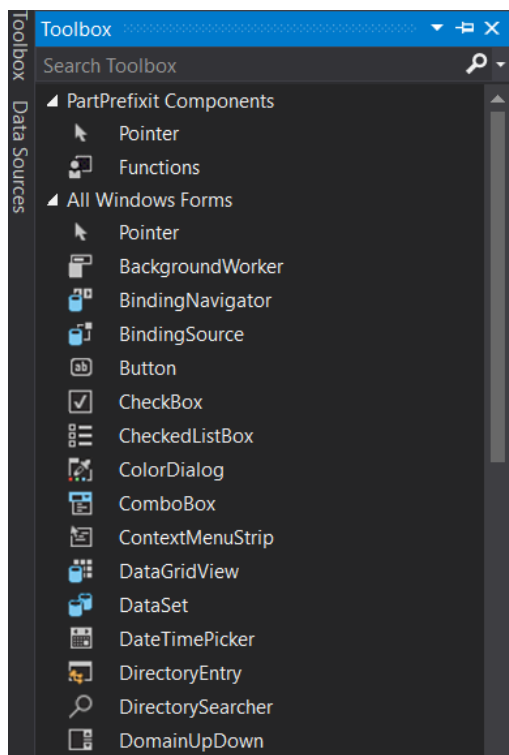
```
private void button5_Click(object sender, EventArgs e)
{
    if (index > 0)
    {
        index--;
    }
    else
    {
        index = puuttuvat.Count - 1;
    }
    ...
}
```

Ohjelmakoodin rakenteesta pyritään tekemään mahdollisimman selkeä muun muassa hyödyntämällä funktioita usein toistuville komennoille. Ohjelmaa kommentoidaan runsaasti, jotta se olisi paremmin ymmärrettävissä. Kommentointi tapahtuu joko // -merkeillä, jolloin jokaisen rivin alkuun täytyy lisätä kyseiset merkit. Vaihtoehtoisesti useamman rivin kommentointi tapahtuu /* ja */ -kommenttimerkkien väliin.

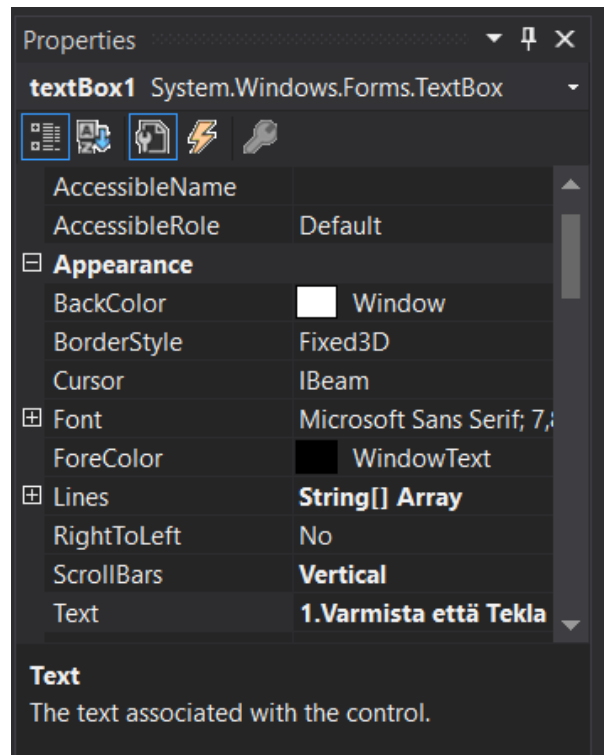
8 KÄYTTÖLIITTYMÄ

Käyttöliittymästä pyritään tekemään mahdollisimman selkeä ja johdonmukainen. Käyttöliittymään tehdään selkeät ohjeistukset, kuinka sitä käytetään. Huomioidaan ikkunan sopiva koko, painikkeiden määrä ja toiminnot. Asettelu tehdään niin, että se on toimiva ja tarkoituksenmukainen ja painikkeet sidotaan ikkunaan kiinni, jolloin ne pysyvät järkevällä paikalla ikkunaa pienentäessä. Lopuksi lisätään ohjelmaan tekijänoikeusmerkintä ja logo.

Visual Studio tarjoaa monipuoliset työkalut käyttöliittymien graafisiin muokkauksiin (Kuvio 6). Tässä käyttöliittymässä on hyödynnetty tekstikenttiä (textbox), tekstikenttiä selittäviä nimiöitä (label), painikkeita (button) sekä valintapalloja (radioButton). Valitun työkalun muokkausasetuksia käyttöliittymä voidaan esimerkiksi nimetä halutuksi sekä muuttaa tekstikenttä selattavaksi (textbox.Multiline). Kuviossa 7 näkymä käyttöliittymän muokkausasetuksista valitulle työkalulle. Tekstikentät ja painikkeet venytetään ja sijoitetaan järkeväksi katsotuille paikoille ja ankkuroidaan sopivaan nurkkaukseen. Käyttöliittymä lukitaan sopivan kokoiseksi, sillä käyttöliittymää ei ole tarvetta suurentaa.



Kuvio 6. Työkalunäkymä



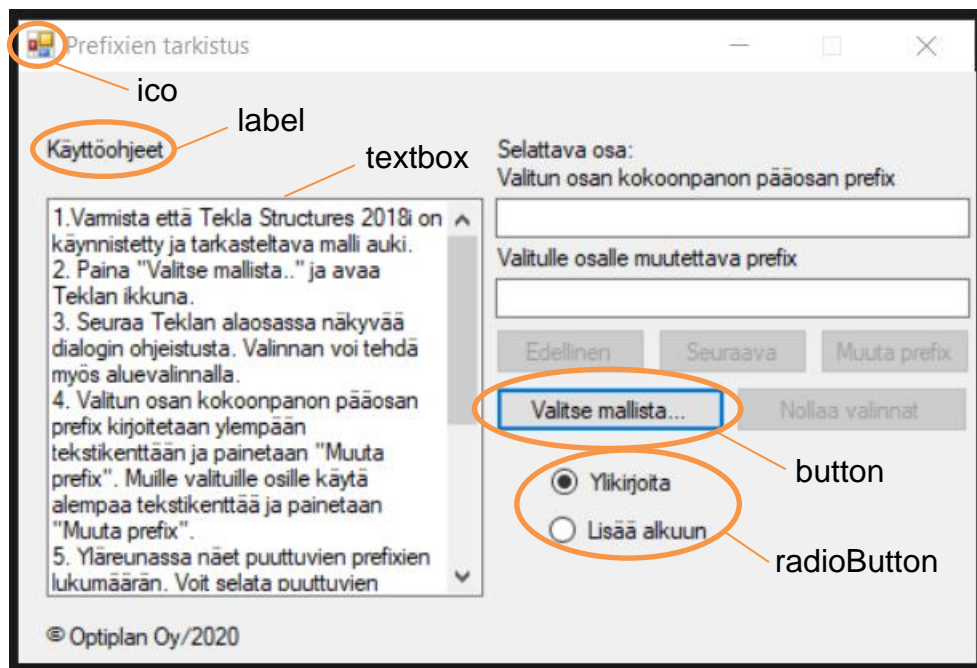
Kuvio 7. Muokkausasetusnäkymä

Tekstikenttien (textbox) lisäyksellä ohjelmassa on mahdollisuus pakottaa valitulle objektille prefix käyttäjän valinnan mukaan. Valuosiens prefixejä korjattaessa voidaan käyttää valintapalloja. Valmiiksi kirjoitettu tekstikentän sisältö voidaan joko ”lisätä alkuun”, jolloin sisältö ”EB_VALU” tulee prefixin nimen eteen. Vaihtoehtoisesti voidaan ”ylikirjoittaa”, eli korvata vanha prefix tekstikentän sisällöllä. Valintapallot eivät voi olla yhtä aikaa valittuna.

Ohjelmaa käynnistäessä tarpeettomat painikkeet himmennetään, kun niille ei ole käyttöä. Ohjelman alussa vain ”Valitse mallista...” -painike näyttäytyy kirkkaana, mutta loput painikkeet näkyvät himmeänä (Kuvio 8). Himmentäminen tehdään *enabled* komennolla.

```
if (puuttuvat.Count != 0)
```

```
{
button4.Enabled = true;
button5.Enabled = true;
button6.Enabled = true;
button7.Enabled = true;
}
```



Kuvio 8. Valmis käyttöliittymä

Käyttöliittymälle lisätään ohjelmaa kuvaava kuvake. Aiheeseen on haastavaa keksiä kuvaavaa kuvaketta, joten lopulliseksi logoksi tulee toimintoa kuvaava

sana (Kuvio 9). Kuvake lisätään ohjelmaan .ico tiedostona, joka on kokoelma kuvia eri resoluutiolla. Logon lisääminen aloitetaan luomalla haluttu kuvake AutoCaddilla, josta se tulostetaan .pdf -tiedostoksi ja muutetaan .ico muotoon.



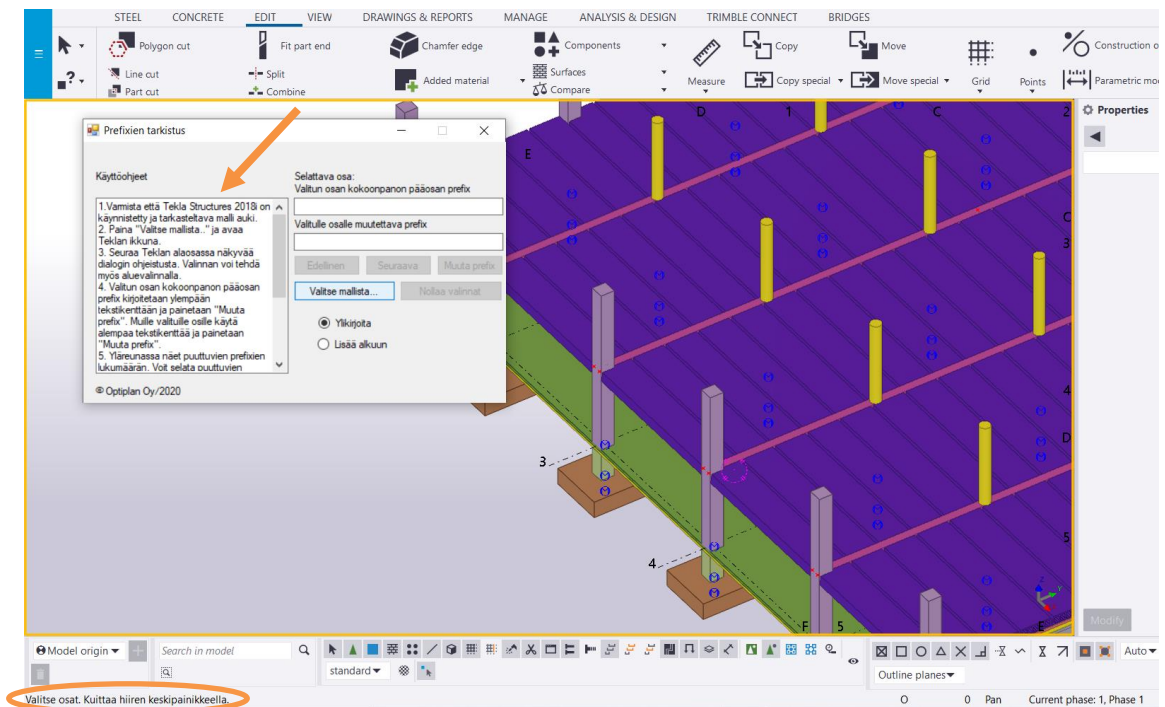
Kuvio 9. Lopullinen logo

Koska ohjelma tulee pelkästään toimeksiantajan käyttöön, lisätään käyttöliittymään tekijänoikeusmerkintänä yrityksen nimi ja vuosiluku. Tekijänoikeusmerkintä lisätään käyttöliittymään "labelina" eli eräänlaisena merkinä, ja sen lisäksi myös itse ohjelman ominaisuuksiin.

9 OHJELMAN KÄYTTÖ

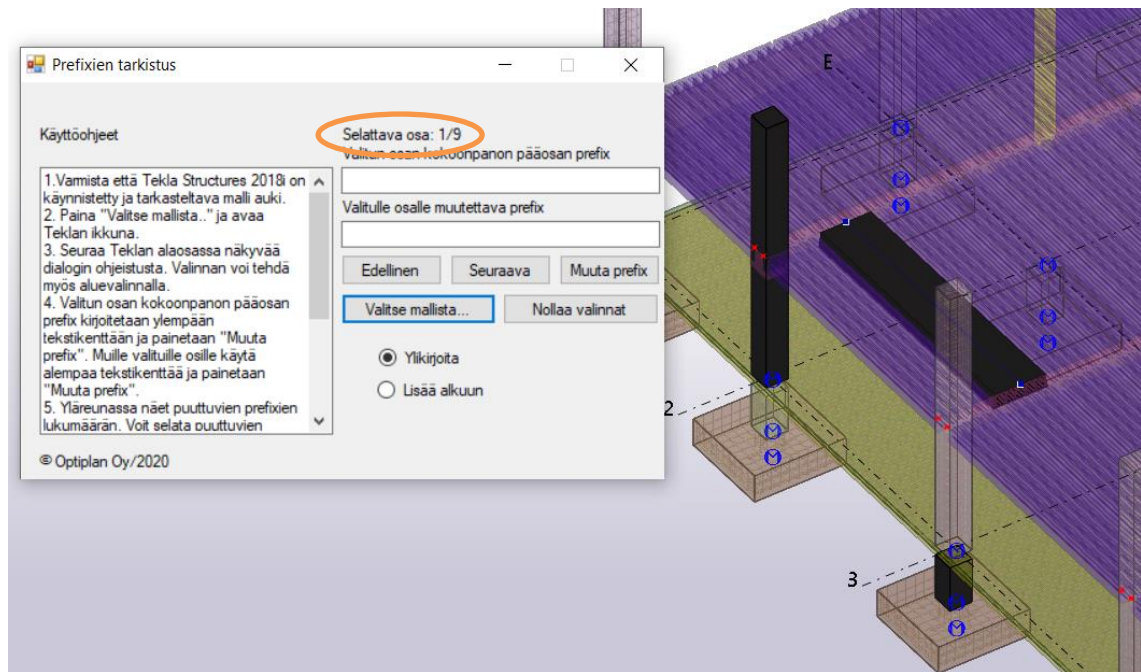
9.1 Ohjelman käyttö vaiheittain

Seuraavassa käydään ohjelman käyttö esimerkkikuvien läpi. Lisäksi pääluvun loppuun on koottu vuokaavio ohjelman käytöstä. Ennen ohjelman käynnistämistä varmistetaan, että tarkasteltava malli on auki. Painetaan painiketta ”Valitse mallista” ja seurataan Teklan alareunan dialogissa näkyvää ohjetta ”Valitse osat. Kuittaa hiiren keskipainikkeella.” Valitaan osat, joilta prefix halutaan tarkistaa, ja painetaan hiiren rullaa. Selattavasta käyttöohjeesta voidaan tarvittaessa katsoa neuvoja (Kuvio 10).



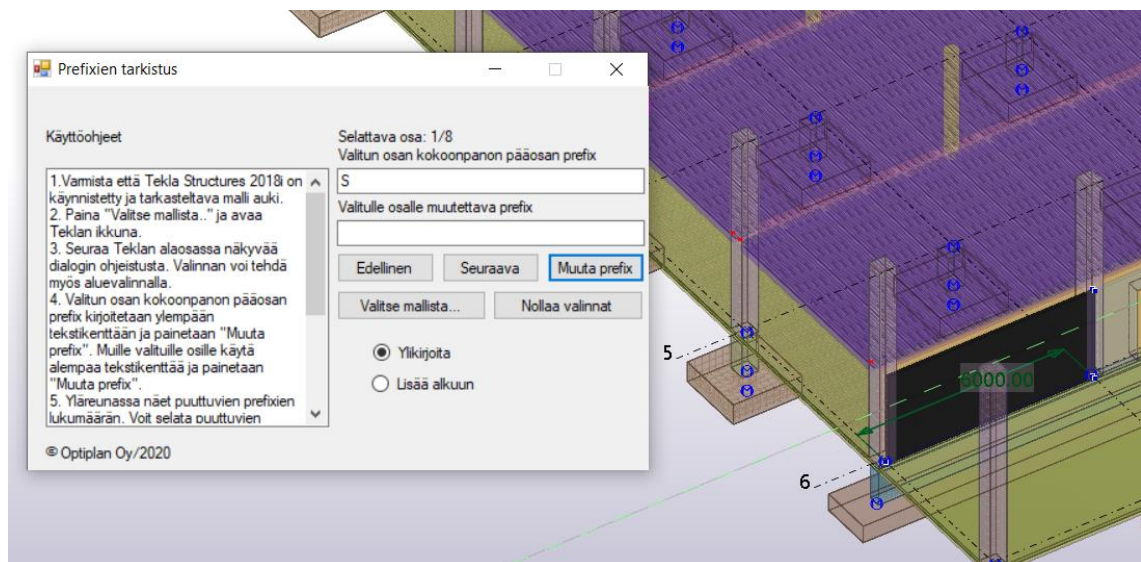
Kuvio 10. Alkutilanne

Ohjelma listaa puutteellisten osien lukumäärän otsikon ”Selattavat osat” perään. Käsiteltävät osat korostuvat mustalla ja muu malli muuttuu läpinäkyväksi (Kuvio 11). Puutteellisten osien listaa voi selata ”Seuraava”- ja ”Edellinen”- painikkeilla. Muutetaan kyseisen osan prefix syöttämällä kenttään numerointisuosituksen (Liite 1) ja elementtitunnusten (Liite 2) mukainen oikea prefix.



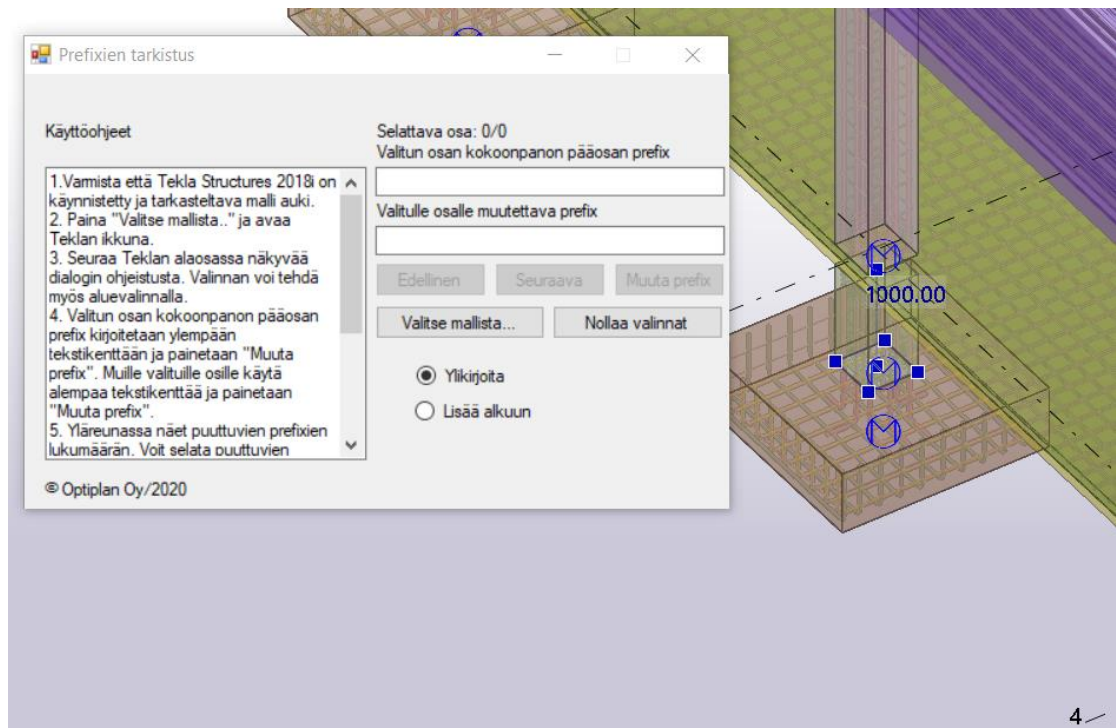
Kuvio 11. Havainnekuva puutteellisten osien erottelusta

Kun ohjelmaan syötetään oikea prefix, ja painetaan "Muuta Prefix", osa palautuu normaalitilaan ja poistuu puuttuvien osien listalta. Ohjelman rajausräjäkymä siirtyy listassa seuraavan korjattavaan osaan (Kuvio 12).



Kuvio 12. Rajausräjäkymän siirtyminen

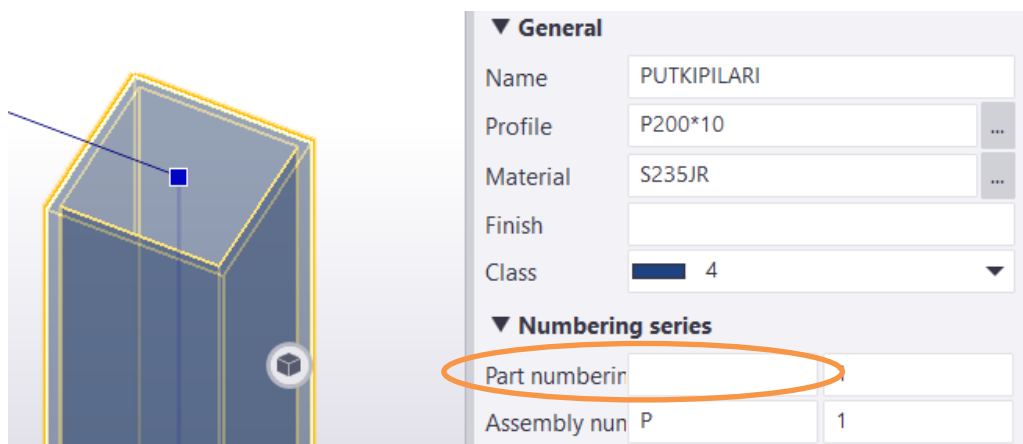
Jatketaan kunnes puutteellisia osia ei enää ole, ja "Selattavien osien" - luettelossa näkyy 0/0 (Kuvio 13). Korjattavia prefixejä ei enää löydy ja ohjelman voi sulkea.



Kuvio 13. Viimeinen selattava osa

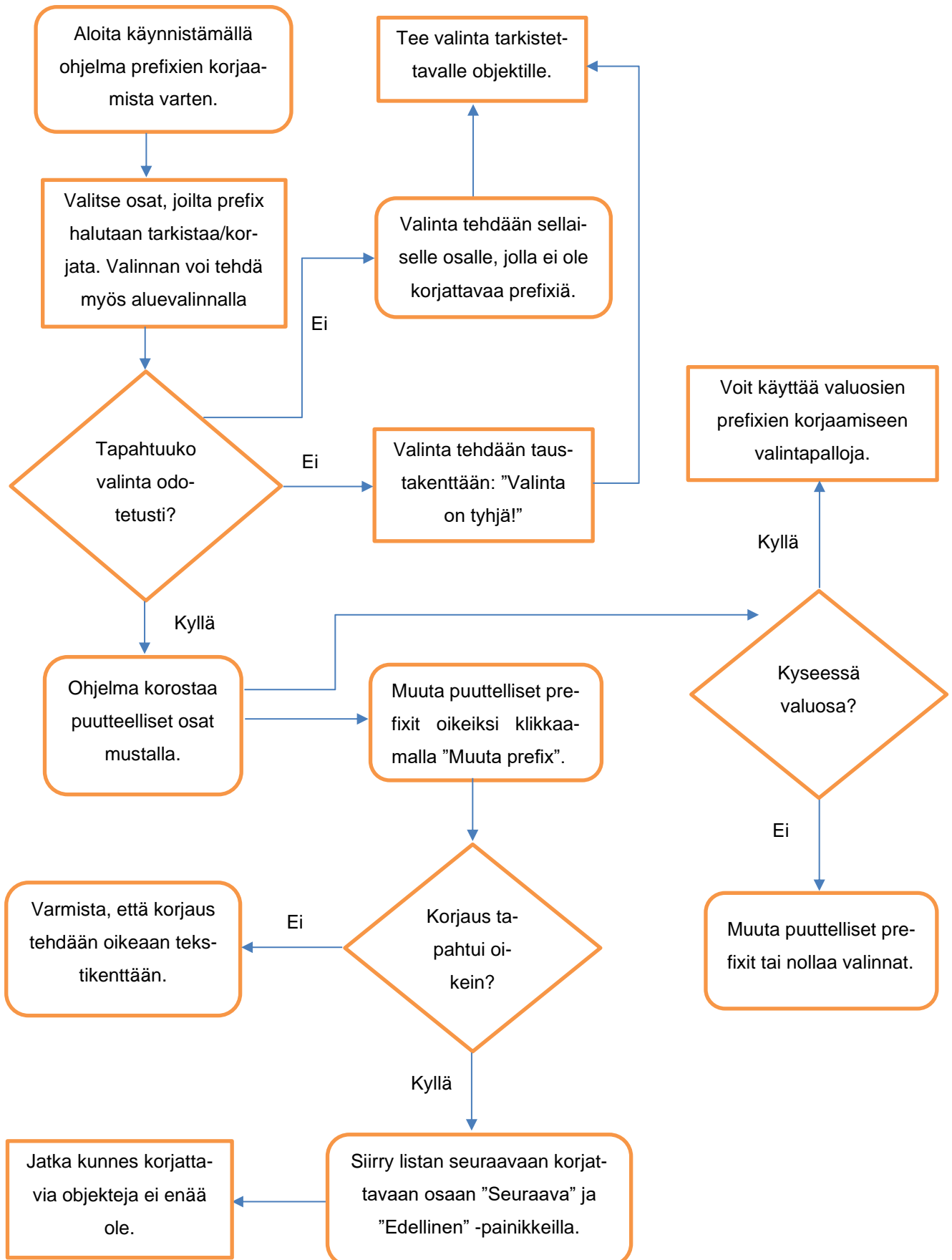
9.2 Huomioita

Ohjelman logiikkaa muutettaessa ohjelma korostaa teräsosien kokoonpanon prefixin ja korjaa sen oikein. Osanumeroinnin puuttumista ohjelma ei kuitenkaan edelleenkään kykene tunnistamaan (Kuvio 14). Näin ollen tämä osa-alue kaipaa yhä lisäkehitystä yrityksen sisäisessä Tekla-ryhmässä. Tämän lisäksi voidaan todeta, että työkalu on todennäköisesti käyttökelpoinen myös kylpyhuoneen syvennyksen vahvikepalkille, sillä työkalu toimii ontelolaatoille oikein.



Kuvio 14. Tilanne, jossa ohjelma ei toimi oikein

9.3 Vuokaavio ohjelman käytöstä



10 TYÖKALUN KÄYTTÖÖNOTTO

Ohjelman on tarkoitus avautua ilman aputiedostoja suoraan Teklassa, ja jotta tämä olisi mahdollista, valmiin toimivan ohjelman lisäksi täytyy muodostaa niin sanottu makro-tiedosto. Makrolla tarkoitetaan automatisoituja asioita, jotka helpottavat käyttäjän toimintaa (Sovelto 2020).

Valmis ohjelma logoineen ja makroineen lisätään Teklaan .exe-tiedostona. Teklan käyttöliittymään eli Ribboniin lisätään oma painike ohjelmalle. Ribbonilla tarkoitetaan Teklan ylälaidan palkissa sijaitsevia painikkeita. Asetuksista löytyvän muokkaustyökalun avulla palkkinäkymää voidaan muokata ja lisätä Teklaan ohjelmia omien tarpeiden mukaan (Trimble 2020a).

Tiedotus ohjelman lisäyksestä Optiplanin rakennesuunnittelijoille tapahtuu yrityksen sisäisen Tekla- kehitysryhmän kautta. Kehitysryhmä kokeilee työkalun toimivuutta ja keskustelee mahdollisesta jatkokehittämisestä. Ohjelmasta säilytetään varmuuden vuoksi myös aikaisemmat varmuuskopiot. Valmis työkalu lisätään Tekla-katalogiin, josta halukkaat yrityksen rakenne- ja elementtisuunnittelijat voivat ladata työkalun käyttöönsä. Ohjelman lisäys vaatii Teklaan ympäristön päivittämisen uusien ominaisuuksien käyttöönottamiseksi.

11 POHDINTA

Aiheena Prefixien tarkistustyökalun kehittäminen Tekla Structuresiin ei ole tavallisimmasta päästä rakennusalalla. Siitä huolimatta, että tietomallinnus on iso osa rakennesuunnittelua, opinnäytetyön painopiste oli ohjelmoinnissa. Opinnäytetyö osoittautuikin erittäin haastavaksi, sillä minulla ei ollut aikaisempaa kokemusta ohjelmoinnista. Aloitin opinnäytetyön tekemisen perehtymällä C# -kieleen Moghadampourin kirjoittaman kirjan avulla. Ohjelmoinnin osaamattomuuden vuoksi minun oli tukeuduttava ohjelmointiosuudessa usein toimeksiantajan ohjaajan puoleen. Uusia oppimiskokemuksia syntyi erityisesti ohjelmoinnista, mutta myös tietomallintamisen puolelta.

Opinnäytetyössä tehtiin selvitystyö toimeksiantajalta saatuun käyttökelpoiseen ohjelmaan, jota jatkokehitettiin tarvittavien osa-alueiden osalta toimivaksi. Selvitystyö tehtiin luotettavaksi tekemällä työkalun testausta luotuihin malleihin useaan kertaan. Yleisesti ottaen opinnäytetyön tavoitteet saavutettiin hyvin. Ainoastaan alikokoonpanojen prefixien osalta ohjelma vaatii yhä edelleen kehittämistä, jotta ohjelma toimii jatkossa kaikissa tapauksissa odotetusti. Opinnäytetyön tuotos auttaa jatkokehittämistä, jonka jälkeen lopullinen työkalu voidaan julkaista Tekla Structuresissa Optiplanin rakenne- ja elementtisuunnittelijoille.

LÄHTEET

ACN ja numerointi 2012. Videonauhoite. Toim. Juha Nieminen. Tekla Structures. Opetusohjelmat. Viitattu 19.2.2020 <https://www.tekla.com/fi/tietoa-meist%C3%A4/webinaarit/video/acn-ja-numerointi>.

Betoniteollisuus ry. 2012. Elementtisuunnittelun mallinnusohje. Viitattu 18.2.2020 [http://www.elementtisuunnittelu.fi/Download/23852/BEC2012%20Elementtisuunnittelun%20mallinnusohje%20\(1\).pdf](http://www.elementtisuunnittelu.fi/Download/23852/BEC2012%20Elementtisuunnittelun%20mallinnusohje%20(1).pdf).

Buildingsmart 2020. Industry Foundation Classes (IFC). Viitattu 14.4.2020 <https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/>.

Dot Net Perls 2020. C# InitializeComponent Method. Viitattu 16.3.2020 <https://www.dotnetperls.com/initializecomponent>.

Elementtisuunnittelu 2020. Mallintava suunnittelu. Viitattu 18.2.2020 <https://www.elementtisuunnittelu.fi/fi/suunnitteluprosessi/mallintava-suunnittelu>.

Geeks for Geeks 2020. Viitattu 12.3.2020 <https://www.geeksforgeeks.org/partial-classes-in-c-sharp/>.

Kuusela, K. 2014. Rakennesuunnittelun tietomallivaatimusten mukaisen komponentin kehittäminen Tekla Structures- ohjelmiston komponenttityökaluilla. Savonia ammattikorkeakoulu. Tekniikan ja liikenteen ala. Opinnäytetyö.

Microsoft 2020. EventArgs Class. Viitattu 16.3.2020 <https://docs.microsoft.com/en-us/dotnet/api/system.eventargs?view=netframework-4.8>.

Microsoft Office 2020. XML- perustietoja aloittelijalle. Viitattu 26.2.2020 <https://support.office.com/fi-fi/article/xml-perustietoja-aloittelijoille-a87d234d-4c2e-4409-9cbc-45e4eb857d44>.

Moghadampour, G. 2012. C# ohjelmointi. Jyväskylä: Sanoma Pro Oy.

Numerointisuositus 2019. Viitattu 18.2.2020 https://moodle.eoppimispalvelut.fi/pluginfile.php/505161/mod_resource/content/1/Numerointisuositus.pdf.

Optiplan Oy 2017. Tietoa meistä. Viitattu 15.1.2020 <https://optiplan.fi/optiplanista/>.

Reviversoft 2020. Viitattu 12.3.2020 <https://www.reviversoft.com/fi/file-extensions/dll>.

RhinoCommon API 2020. Brep Class. Viitattu 5.5.2020 https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Brep.htm.

RT 10-11070 2012. Yleiset tietomallivaatimukset 2012 Osa 5. Rakennesuunnittelu. RT-ohjekortti. Rakennustieto. Viitattu 18.2.2020 <https://www.lapinamk.fi/fi/Opiskelijalle/Sovellukset-ja-pikalinkit,Finna-tiedonhakupalvelu,RT-Net,RT-kortisto>.

RT 10-11080 2012. Yleiset tietomallivaatimukset 2012 Esittely. RT-ohjekortti. Rakennustieto. Viitattu 18.2.2020 <https://www.lapinamk.fi/fi/Opiskelijalle/Sovellukset-ja-pikalinkit,Finna-tiedonhakupalvelu,RT-Net,RT-kortisto>.

RT 10-11209 2016. Yleiset tietomallivaatimukset 2012 Osa 5. Rakennesuunnittelu. Tilaajan ohje. Mallinnustarkkuus. RT-ohjekortti. Rakennustieto. Viitattu 14.4.2020 <https://www.lapinamk.fi/fi/Opiskelijalle/Sovellukset-ja-pikalinkit,Finna-tiedonhakupalvelu,RT-Net,RT-kortisto>.

Sovelto 2020. Makro vai ei? Viitattu 4.5.2020 <https://www.sovelto.fi/makro-vai-ei/>.

Tekla 2020a. Edistysellinen BIM- ohjelmisto rakennesuunnittelun työnkulkuun. Viitattu 14.1.2020 <https://www.tekla.com/fi/tuotteet/tekla-structures>.

- 2020b. Mitä on BIM? Viitattu 18.2.2020 <https://www.tekla.com/fi/tietoa-meist%C3%A4/mit%C3%A4-bim>.

- 2020c. Tekla Open API. Viitattu 14.1.2020 https://teklastructures.support.tekla.com/200/en/sys_tekla_open_api.

Trimble 2020a. Customize the ribbon Viitattu 4.5.2020 https://teklastructures.support.tekla.com/2020/en/gen_customize_ribbon.

- 2020b. Develop applications using Tekla Open API. Viitattu 11.3.2020 https://teklastructures.support.tekla.com/2019/en/sys_tekla_open_api.

- 2020c. Tekla Open API 2018i Reference. Viitattu 17.3.2020 <https://developer.tekla.com/api/6/15509>.
- 2020d. Tekla Open API 2018i Reference. Viitattu 17.3.2020 <https://developer.tekla.com/api/6/15506#!>.
- 2020e. Tekla Open API 2019i Reference. Viitattu 16.3.2020 <https://developer.tekla.com/tekla-structures/api/9/15524>.
- 2020f. Tekla Open API 2019i Reference. Viitattu 17.3.2020 <https://developer.tekla.com/tekla-structures/api/9/14457>.
- 2020g. Tekla Open API 2020 Reference. Viitattu 18.3.2020 <https://developer.tekla.com/tekla-structures/api/10/14386>.

W3Schools 2020. XML Soap. Viitattu 26.2.2020
https://www.w3schools.com/xml/xml_soap.asp.

Visual Studio 2020. Best-in-class tools for any developer. Viitattu 14.1.2020
<https://visualstudio.microsoft.com/>.

LIITTEET

Liite 1. Numerointisuositus, taulukko

Liite 2. Elementtitunnukset, taulukko

Nimi	Name	Class	Assembly	Part	Plate	Rakenneosa	Structure	TERÄS / STEEL Oletusosa / default profile
PYSTYRAKENTEET	VERTICAL STRUCTURES	1-20	Prefix and start number					
VALSSATTU PILARI	ROLL COLUMN	1	C1	P1	L1	Valssattu pilari	Hot rolled column	HEA300
WI PILARI	WI COLUMN	2	WC1	WP1	L1	Hitattu I-pilari	Welded I/H column	WI400-10-15x200
WP PILARI	WB COLUMN	3	WC1	WP1	L1	Hitattu koteloilari	Welded box column	WE400-20-20x400/25
PUTKIPILARI	PIPE COLUMN	4	C1	P1	L1	Putkipilari, pyöreä tai suor	Pipe column, round or re	PF200*200*10
SEINÄ SIDE	VER BRACE	5	V1	VP1	L1	Seinäside	Vertical brace	CFRHS200*5
SEINÄ WB SIDE	VER WB BRACE	6	WV1	VP1	L1	Hitattu koteloseinäside	Welded box brace	WE400-20-20x400/25
KEHYSLELEMENTTI	FRAME	7	F1	FP1	L1	Ikkuna-, ovi- tai aukkokehys	Window, door or opening frame	CFRHS100*4
EXTRA_OSA	EXTRA PART	20	EX1	EXP1	EXL1	Extra-osa	Extra part	
VAAKARAKENTEET	HORIZONTAL STRUCTURES	21-40						
SEKUNDAARI PALKKI	SEC. BEAM	21	SB1	P1	L1	Sekundääripalkki	Secondary beam	IPE200
PRIMAARI PALKKI	PRIM BEAM	22	B1	P1	L1	Primääripalkki	Primary beam	IPE300
WI PALKKI	WI BEAM	23	W1	WP1	L1	Hitattu I-palkki	Welded I-beam	WI400-8-15x200
WI_KATTILAPALKKI	WI BOILER BEAM	24	W1	WP1	L1	Hitattu I-palkki kattilapalkkina	Welded I-beam in a boiler building	WI1000-20-25x300
WB_KATTILAPALKKI	WB BOILER BEAM	25	W1	WP1	L1	WB-palkki kattilapalkkina	Welded box beam in a	WB1000-15-20x300/25
TASO SIDE	HOR. BRACE	26	H1	1	L1	Tasoside	Horizontal brace	CFRHS150*5
NOSTINPALKKI	HOIST BEAM	27	HB1	P1	L1	Hitattu tai valssattu I-palkki	Welded or hot rolled I-be	HEB300
NOSTURIRATAPALKKI	CRANE BEAM	28	HR1	HP1	HPL1	Nosturiratapalkki	Crane beam	HEB300
HQ_PALKKI	HQ BEAM	29	HQ1	HQP1	WOL1	HQ-palkki	HQ beam	HQ320-5-15x190-15x500
WQ_PALKKI	WQ BEAM	30	WQ1	WQP1	WOL1	WQ-palkki	WQ beam	WQ320-5-15x190-15x500
DELTA_PALKKI	DELTA BEAM	31	D1	DP1	DL1	Delta-palkki	Deltabeam (Peikko)	D32-300
KVATRO_PALKKI	KVATRO BEAM	32	KV1	KVP1	KVPL1	Kvatro-palkki	Kvatro beam (Teräselementti)	
KYNNELLEVY	CHEC. PLATE	33	CP1	EXP1	CPL1	Kynnellevy	Steel plate with checked pattern	
EXTRA_OSA	EXTRA PART	34	EX1	EXP1	EXL1	Extra-osa	Extra part	
RISTIKORAKENTEET	TRUSS STRUCTURES	16						
RISTIKKO	TRUSS	16	TR1	TP1	TL1	Ristikko	Truss	
ASENNUSOSAT	ERECTION PARTS	17						
ASENNUSLEVY	EREC. PLATE	17	EPL1		EL1	Yksittäisiä asennusosia. Käytetään mahd. ilman kokoonpanotunnusta, koska ei asetettavissa kaikissa systeemiliitoksissa.	Loose parts used in erection of a building. Possibly without assembly prefix as they cannot be set on some system connections.	
ASENNUSOSA	EREC. PART	17	EP1	EP1				
L-TERAKSET (LIITOKSISSA)	CLIP ANGLES (IN CONNECTIONS)	18						
L_TERÄS	CLIP ANGLE	18	X1	XP1	XP1	L-teräs liittoksesta	Clip angle	
PORTAAT JA TIKKAAAT	STAIRS AND LADDERS	41-50						
PORRAS	STAIR	41	SS1	SSP1	SSL1	Porrasreislankku	Load bearing stair beam	UNP160
ASKELMA	STEP	42	ST1	STP1	STL1	Porrasaskelma	Stair step	

KÄSIKAIDE	RAILING	43	EHRT	EHRP1	EHRL1	Elementtikaide	Stair assembly	
KÄSIKAIDE	RAILING	44	HR1	HRP1	HRL1	Intokaide	Stair assembled at construction site	
TIKAS	LADDER	45	LRT	LRP1	LRL1	Tikas	Ladder	
VALUTARIVIKKEET								
VALUTARIVIKE	EMBED	99				Valutarivike	Embedded	
VALUTARIVIKE	EMBED	100				Vakio valutarivike	Standard embed	
VALUTARIVIKE	EMBED	101	EX1	EXP1	EXL1	Erikois valutarivike	Special embed	
KUORIRAKENTEET								
POIMULEVY	CORRUGATED_SHEET	102				Poimulevy	Corrugated sheet	
FASETTI	FACET	103	CS1	CSP1	CSP1	Rannila Fasetti	Facet (Rannila)	
KASETTI	CASSETTE	104	CA1	CAP1	CAP1	Rannila Liberta	Liberta (Rannila)	
SW_PANEELI	SW_PANEL	105	SP1	SP1	SP1	Rannila Panel 3lock	Panel 3lock (Rannila)	
KASETTI	CASSETTE	106	LT1	LT1	LT1	Rannila Casetti	Casette (Rannila)	
ORSI	PURLIN	107	PU1	PUP1	PUP1	Teräselementti Kvatro	Kvatro (Teräselementti)	
NORDICON	NORDICON	108	NO1	NOP1	NOP1	C-, H- tai Z-orsi	Cold rolled purlin	
PELTI	SHEET	109	FL1	FLP1	FLP1	TC-, TU-, tai TUL-Nordicon	TC, TU or TUL purlin (Ruukki Nordicon)	
K_POIMULEVY	BEARING_SHEET	110	CS1	CSP1	CSP1	Kantava poimulevy	Load bearing corrugated sheet	
SW_PANEELI	SW_PANEL	111	CE1	CEP1	CEP1	Teräselementti Kvatro	Kvatro panel (Teräselementti)	
BETONIELEMENTIT / CONCRETE ELEMENT								
Nimi	Name	Class	Cast Unit	Part	Plate	Rakennesa	Structure	Oletusosa / default profile
Runkoelementit	Framework elements	201-209	Prefix and start number					
PILARI	COLUMN	201	P1			Suorakaidepilari	Rectangle beam	380*380
PILARI	COLUMN	202	P1			Pylvöreä pilari	Round column	D380
SUORAKAIDEPALKKI	RECTANGLE BEAM	203	JK1			Jännäbetonipalkki, suorakaide	Rectangle beam, prestressed	780*380
SUORAKAIDEPALKKI	RECTANGLE BEAM	204	K1			Suorakaidepalkki	Rectangle beam	780*380
LEUKAPALKKI	GNATHIC BEAM	205	JK1			Jännitetty leukapalkki	Gnathic beam, prestressed	RCL300*600-400*150
LEUKAPALKKI	GNATHIC BEAM	206	K1			Leukapalkki	Gnathic beam	RCL300*600-400*150
MAATALALEUKAPALKKI	LOW_GNATHIC BEAM	207	JK1			Matalaleukapalkki	Gnathic beam, low height	RCD1280*375*580*100*100
HI-PALKKI	RIDGE 1-BEAM	208	HT1			Hi-palkki	Ridge 1-beam	
I-PALKKI	I-BEAM	209	IT			I-palkki	I-beam	
Laattelementit		210-215						
ONTELOLAATTA	HOLLOW_CORE SLAB	210	OL1			Ontelolaatta	Hollow core slab	P27(265X1200)
KUORILAATTA	FLOOR PLANK	211	KL1			Kuorilaatta	Floor plank below a cast in situ slab	KL100(100X1200)
TT-LAATTA	TT-SLAB	212	TT1			TT-laatta	TT-slab	TT350*356-120-50-1460-
HT-LAATTA	RIDGE TT-SLAB	213	HTT1			HTT-laatta	Ridge TT-slab	0.03-150-0.25

PORRASLAATTA	STAIR SLAB	214	L1			Porraslaatta	Stair slab	260*3000
LEPOTASOLAATTA	LANDING	215	L1			Lepotasolaatta	Landing	260*3000
TEK-LAATTA	TEK-SLAB	216	TEK1			TEK-Laatta	TEK-slab	
Seinäelementit								
	Wall elements	220-229						
SANDWICH	SANDWICH	220	R1			Ei-kantava, sandwich	Non bearing sandwich	
SANDWICH	SANDWICH	221	S1			Kantava, sandwich	Bearing sandwich	
SOKKELI	SOCLE	222	AR1			Ei-kantava, sokkeili	Non bearing socle	
SOKKELI	SOCLE	223	AS1			Kantava, sokkeili	Bearing socle	
SOKKELI	SOCLE	224	AV1			Sokkeili	Socle	
ULKOKUORI	OUTER SHELL	225	KE1			Ulkokuori	Outer shell	2985*70
SISÄKUORI	INNER SHELL	226	RK1			Ei-kantava, sisäkuori	Non bearing inner shell	2950*80
SISÄKUORI	INNER SHELL	227	SK1			Kantava, sisäkuori	Bearing inner shell	2450*180
VALSEINA	INTERNAL WALL	228	V1			Valseinä	Internal wall	2685*200
MAANPAINESEINÄ	GROUND PRESSURE WALL	229	MP1			Maanpaineseinä	Ground pressure wall	2685*200
Parveke-elementit								
	Balcony elements	250-						
PARVEKELAATTA	BALCONY SLAB	250	CL1			Parvekelaaatta	Balcony slab	
PARVEKEPILARI	BALCONY COLUMN	251	CP1			Parvekepilari	Balcony column	D250
PARVEKEPIELI	BALCONY WALL	252	M1			Parvekepieili	Balcony wall	2985*180
PARVEKEKAIDE	BALCONY RAILING	253	Z1			Parvekekaide	Balcony railing	1080*100
PARVEKEKATTOLAATTA	BALCONY ROOF	254	CL1			Parvekkeen kattolaatta	Balcony roof slab	
Mut-elementit								
	Other elements	260-						
PAALU	PILE	260	PA1			Paalu	Pile	
HISSIKUILU	ELEVATOR SHAFT	261	HKJ1			Hissikuilu	Elevator shaft	
HISSIKATTO	ELEVATOR ROOF	262	HKA1			Hissikatto	Elevator roof	
HISSIPOHJA	ELEVATOR FLOOR	263	HPO1			Hissipohja	Elevator floor	
PORRASELEMENTTI	STAIR	264	PO1			Porraslementti	Stair element	

Nimi	Name	Class	Cast Unit	Part	Plate	Rakenneosa	Structure	PAIKALLAVALU / CAST IN SITU
Perustukset								
	Foundations	310-						
ANTURA	FOOTING	302	PV-A1			Antura	Pad footing	1500*1500
PAULUANTURA	PILE FOOTING	303	PV-PA1			Pauluantura	Pile cap footing	1500*1500
PERUSPILARI	FOUNDATION COLUMN	304	PV-PP1			Peruspilari	Foundation column/pillar	480*480
PERUSMUURI	FOUNDATION WALL	305	PV-PM1			Perusmuuri	Foundation wall	300*600
SOKKELPALKKI	SOCLE	306	PV-SP1			Sokkeilipalkki	Socle beam	300*600
KONEPERUSTUS	MACHINE FOUNDATION	307	PV-KP1			Koneperustus	Machine foundation in a factory	
Runkorakenteet								
	Framework structures	320-						
PILARI	COLUMN	320	PV-P1			Pilari	Column	480*480
PALKKI	BEAM	322	PV-K1			Palkki	Beam	780*380
LAATTA	SOLID SLAB	323	PV-L1			Laatta	Slab	200
LEPOTASOLAATTA	LANDING	324	PV-L1			Lepotasolaatta	Landing	250
MAANVARAINENLAATTA	GROUND SLAB	325	PV-ML			Maanvarainenlaatta	Solid slab on ground	100
PINTALAATTA	SURFACE SLAB	326	PV-PL			Pintalaatta	Solid slab on a plank or hollow core slab	60

Liite 1 4(5)

SEINÄ	WALL	324	PV-V1		Seinä	Solid wall	2685-200
Litttorakenteet	Composite structures	330-					
LITTOPIILARI	COMPOSITE COLUMN	330	LP1		Littopilari	Composite column	
LITTOPALKKI	COMPOSITE BEAM	331	LK1		Littopalkki	Composite beam	
LITTOLAATTA	COMPOSITE SLAB	332	LT1		Littolaatta	Composite slab	
Muut rakenteet	Other structures	340-					
TILISEINÄ	BRICK WALL	340	M-TS1		Tiliseinä	Brick wall	
HARKKOSEINÄ	BLOCK WALL	341	M-HS1		Harkkoseinä	Block wall	
ERISTE	INSULATION	342	ER1		Eriste	Thermal insulation	
KANAALISEINÄ	CHANNEL WALL	343	PV-KAS1		Kanaalin seinä	Channel wall	
KANAALIPOHJA	CHANNEL FLOOR	344	PV-KAP1		Kanaalin pohja	Channel floor	
PORRAS	STAIR	345	PV-PO1		Porras	Stair	
PORRASHUONE	STAIR ROOM	346	PV-PH1		Porrashuone	Stair room	
HIISKIKULU	ELEVATOR SHAFT	347	PV-KH1		Hiiskikulu	Elevator shaft	
MUOTTIPELTI	MOULD SHEET	348	MUP1		Muottipeitti	Corrugated sheet working as a slab mould	

Nimi	Name	Class	Cast Unit	Part	Plate	Rakenneseosa	Structure	KATKOT MATERIAALI / ALL MATERIALS
Valmiit rakenteet	Existing structures	400-	Prefix and start number					Oletusosa / default profile
NYKYINEN	EXISTING	400	NYK1	NYK1	NYK1	Olemassaoleva	Existing structure	
PURETTAVA	TO_BE_DEMOLISHED	401	PUR1	PUR1	PUR1	Purettava rakenne Esimerkiksi toisen yrityksen suunnittelema rakenne, joka esitetään referenssinä	Structure to be demolished	
REFERENSSI	REFERENCE	402	REF1	REF1	REF1		Reference structure (e.g. structure designed by another company)	

Elementit	Nimi	Name	Class	Rebar		Rakennosa	Structure	REINFORCEMENT
raudoliteet	Element reinforcement		500-					
PAATERAS	MAIN_BAR	500	" /1"			Paäteräs	Main bar	20
HAKA	STRIRUP	501	" /1"			Haka	Strirup	8
RIPUSTUSTERÄS	HANG_BAR	502	" /1"			Ripustusteräs	Hang bar	8
NOSTOLENKKI	LIFTING_ANCHOR	503	" /1"			Nostolenkki	Lifting anchor	φ12
TARTUNTA	DOWEL	504	" /1"			Tartunta	Dowel	16
JÄÄNEPUNOS	STRAND	505	" /1"			Jäänepunos	Strand	12,6
erilliset rakennosat esim. anturaut	Separate structures e.g. footings							
520-								
PAATERÄS	MAIN_BAR	520	Rp1			Paäteräs	Main bar	20
HAKA	STRIRUP	521	RH1			Haka	Strirup	8
RIPUSTUSTERÄS	HANG_BAR	522	RR1			Ripustusteräs	Hang bar	8
NOSTOLENKKI	LIFTING_HOOK	523	RN1			Nostolenkki	Lifting hook	φ12
TARTUNTA	DOWEL	524	RT1			Tartunta	Dowel	16
SISÄTERÄS	INSIDE_BAR	525	RS1			Sisäteräs	Inside Bar	20

ULKOTERÄS	OUTSIDE_BAR	526	RU1				Ulkoteräs	outside_bar	20
LISÄTERÄS	SIDE_BAR	527	RL1				Lisäteräs	Side_bar	20
Palkkialavaiun raudoitteet	Cast in situ reinforcement	540-							
YPY	UST	540	YPY1				Yläpinnan ylempi teräs	Upper surface top bar	12
YPa	usb	541	YPa1				Yläpinnan alempi teräs	Upper surface bottom bar	12
APY	IST	542	APY1				Alapinnan ylempi teräs	Lower surface top bar	12
APa	ISb	543	APa1				Alapinnan alempi teräs	Lower surface bottom bar	12
Verkot	Meshes	560-							
YPSUORAKAIDE	USRECTANGLE	561	SV1				Suorakaideverkko	Rectangle mesh	8/8 150/150
apsuorakaide	ISRECTANGLE	561	SV1				Suorakaideverkko	Rectangle mesh	8/8 150/150
TAIVUTETTU	BENT	561	TV1				Taivutettu verkko	Bent mesh	8/8 150/150
YPMONIKULMIO	USPOLYGON	561	MV1				Monikulmio	Polygon	8/8 150/150
apmonikulmio	ISPOLYGON	561	MV1				Monikulmio	Polygon	8/8 150/150

Elementtitunnukset

10.2.2011

Elementtisuunnitelmiin nimetään erilaiset elementtityypit tunnuksella, joka koostuu tyyppin mukaisesta kirjainyhdistelmästä ja elementtityypin yksilöivästä numerosta esim. RK-1. Taulukossa on esitetty eri elementtityyppien kirjaintunnukset. Samanlaisia elementtejä voi kohteessa olla useampia kappaleita. Elementeille suositellaan käytettäväksi erillistä juoksevaa ID- numerointia, jolla yksilöidään yksittäinen elementti.

Taulukko. Elementtitunnukset

ELEMENTTITYYPPI	ELEMENTTI	TUNNUS
PERUSTUSELEMENTIT	ANTURAELEMENTTI	A
	PILARIHOLKKIELEMENTTI	PH
	SOKKELIELEMENTTI (EI KANTAVA)	AN
	SOKKELIELEMENTTI (KANTAVA)	AS
	SOKKELIPALKKI	AK
	SOKKELIRUUTUELEMENTTI (MAANPAINE)	AR
	SOKKELIELEMENTTI (MAANPAINE, YKSI KUORI)	AV
	TUKIMUURIELEMENTTI	TKE
PILARIELEMENTIT	PILARI	p ⁽¹⁾
SEINÄELEMENTIT	VÄLISEINÄ	V
	VÄLISEINÄ (SEINÄMÄINEN PALKKI)	VSP
	RUUTUELEMENTTI (KANTAVA)	S
	RUUTUELEMENTTI (EI KANTAVA)	R
	SISÄKUORIELEMENTTI (KANTAVA)	SK
	SISÄKUORIELEMENTTI (EI KANTAVA)	RK
	SISÄKUORIELEMENTTI (KANTAVA, ERISTE+RAPPAUS)	SKR
	SISÄKUORIELEMENTTI (EI KANTAVA, ERISTE+RAPPAUS)	RKR
	NAUHAELEMENTTI (KANTAVA)	NK
	NAUHAELEMENTTI (EI KANTAVA)	N
	KUORIELEMENTTI	KE
PALKKIELEMENTIT	PALKKIELEMENTTI (TERÄSBETONI)	K
	JÄNNEBETONIPALKKI (I-PROFIILI)	I
	JÄNNEBETONIPALKKI, (HI-PROFIILI)	HI
	JÄNNEBETONIPALKKI, (MUUT PROFIILIT)	JK ⁽²⁾
LAATTAELEMENTIT	LAATTAELEMENTTI (MASSIIVILAATTA, VÄLIPOHJA)	L
	ALAPOHJALAATTA (MASSIIVILAATTA, ERISTETTY)	EL
	JÄNNITETTY LAATTAELEMENTTI	JL
	ONTELOLAATTA	O ⁽³⁾
	ONTELOLAATTA (LÄMPÖERISTETTY)	O ⁽³⁾
	ONTELOLAATTA (REI90-PALOLAATTA)	15O
	ONTELOLAATTA (REI120-PALOLAATTA)	2O
	ONTELOLAATTA (YLÄPUNOSLAATTA)	YO
	ONTELOLAATTA (KYLPHYUNELAATTA)	OK ⁽⁴⁾
	KUORILAATTA	KL
	TT-LAATTA	TT
	HTT-LAATTA	HTT

PARVEKE-ELEMENTIT	PARVEKE- ELEMENTTI	C
	PARVEKELAATTAELEMENTTI	CL
	JÄNNITETTY PARVEKELAATTAELEMENTTI	JCL
	PARVEKEPIELIELEMENTTI	M
	PARVEKEKAIDE- ELEMENTTI	Z
	PARVEKKEEN KATTOELEMENTTI	CX
	JÄNNITETTY PARVEKKEEN KATTOELEMENTTI	JCX
PORRASELEMENTIT	PORRASELEMENTTI	T
HISSIKUILUN ELEMENTIT	HISSIKUILUELEMENTTI	HK ⁽⁵⁾
	HISSIKUILUN POHJAELEMENTTI	HKA
	HISSIKUILUN YLÄPÄÄN ELEMENTTI	HKY
ERIKOISELEMENTIT	HORMIELEMENTTI	H
	ERIKOISKAPPALE	..X ⁽⁶⁾

- (1) Jos kohteessa on useampia pilarityyppejä, kannattaa käyttää erilaista tunnusta erityyppisille elementeille.
- (2) Tunnuksella JK voidaan tyypittää jännitettävät suorakaide-, leuka- ja ristipalkit.
- (3) Ontelolaatan tunnukseen sisällytetään myös laattatyyppin korkeuden erittelevä numerotunnus. Esimerkiksi ontelolaatta 370mm korkea, tunnus O37-. Eri laattavalmistajilla on käytössä omat etuliitteensä.
- (4) Esimerkiksi 320mm korkea kylpyhuoneontelolaatta, tunnus O32K-.
- (5) Tunnukseen voidaan lisätä elementin muodon kertova tunniste. Esimerkiksi HKU on tasossa U:n muotoinen elementti ja HKL on L:n muotoinen elementti.
- (6) Tavallisesta poikkeavien elementtien tunnukseen lisätään merkintä X. Esimerkiksi; rakennuskohteessa on yksi laattaelementti, jonka paksuus on poikkeava. Poikkeavan elementin tunnus on LX-.