OULUN AMMATTIKORKEAKOULU

Robin Huumonen

DSI-TIEDOSTOJEN LUKEMINEN CWISE-JÄRJESTELMÄÄN

DSI-TIEDOSTOJEN LUKEMINEN CWISE-JÄRJESTELMÄÄN

Robin Huumonen
Opinnäytetyö
Syksy 2022
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, laite- ja tuotesuunnittelun suuntautumisvaihtoehto

---

Tekijä: Robin Huumonen
Opinnäytetyön nimi: DSI-tiedostojen lukeminen CWise-järjestelmään
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Syksy 2022 Sivumäärä: 25

Opinnäytetyö tehtiin kehittämistyönä yritykseen Wisetime Oy. Työn päätavoitteena oli lukea johdinsarjoja kuvaavia DSI-tiedostoja toiminnanohjausjärjestelmän tietokantaan. Lisäksi tiedostoista saatuja tietoja jatkokäsiteltiin asiakkaan tarpeita varten ja käyttöliittymään kehitettiin työkalu eri DSI-tiedostojen vertailuun.

Työ tehtiin päivätyön yhteydessä ja on osa laajempaa projektia. Työ aloitettiin sisäisellä palaverilla. Säännölliset palaverit asiakkaan kanssa aloitettiin, kun DSI-tiedostoja saatiin luettua CWise-järjestelmään. Lukuohjelma kehitettiin DSI:n määrittelydokumentin mukaisesti. Myös joitain valmistajakohtaisia DSI:n määrittelyn ulkopuolisia käytänteitä huomioitiin. Työn palvelinohjelmisto kehitettiin Oraclen tietokannassa ja käyttöliittymä Delphi-ohjelmankehitysympäristössä.

Työn tuloksena CWise-järjestelmään saatiin kehitettyä tavoitteiden mukaisesti ominaisuus DSI:den lukemiseen ja vertailuun. Lisäksi kehitettiin ominaisuus päivittää CWisen tuoterakennetta DSI-tiedostosta löytyvillä komponenteilla ja koottiin johdinsarjakohtaiset tiedot erikseen usean sarjan sisältävästä tiedostosta.

Asiasanat: toiminnanohjausjärjestelmät, tietokantaohjelmat, DSI-tiedostomuoto

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Device and Product Design

---

Author: Robin Huumonen
Title of thesis: Importing DSI files into CWise
Supervisor: Kari Jyrkkä
Term and year when the thesis was submitted: Autumn 2022
Number of pages: 25

The thesis was done as a development to Wisetime Ltd. The main objective was to import DSI files into enterprise resource planning system's database in a relational format. DSI is a file structure for representing wire harness in plain text. The second objective was to create a comparison tool between two DSI imports.

Development was a part of a larger customer project. Import tool was developed using the official specification from Siemens EDA. Some manufacturer specific implementations were also noted. Backend was developed in Oracle database and user interface with Delphi.

As a result, the import and comparison tools were developed and released to the customer for testing. Objectives set for this thesis were achieved. Work around imported DSI data continues; later it will be used to create items automatically in centralized ERP.

Keywords: enterprise resource planning systems, database programs, DSI file format

SISÄLLYS

1 JOHDANTO

Opinnäytetyö tehtiin yritykseen Wisetime Oy. Wisetime on noin kolmikymmenvuotias oululainen ohjelmistotalo. Heidän päätuotteensa on Wise-toiminnanohjausjärjestelmä. Järjestelmä sisältää kaikki teollisuusyrityksen tarvitsemat liiketoimintaprosessin ohjauksen osa-alueet. (1.)

Opinnäytetyön aiheena oli DSI-tiedostojen lukeminen CWise-järjestelmään. DSI on Mentor Graphicsen, sittemmin Siemens EDAn tiedostomuoto (2), joka kuvaa johdinsarjoja tekstimuodossa. Tiedosto tuotetaan yleensä tietokoneavusteisella suunnitteluohjelmalla. Formaatti on usean tunnetun kuorma-autovalmistajan käytössä.

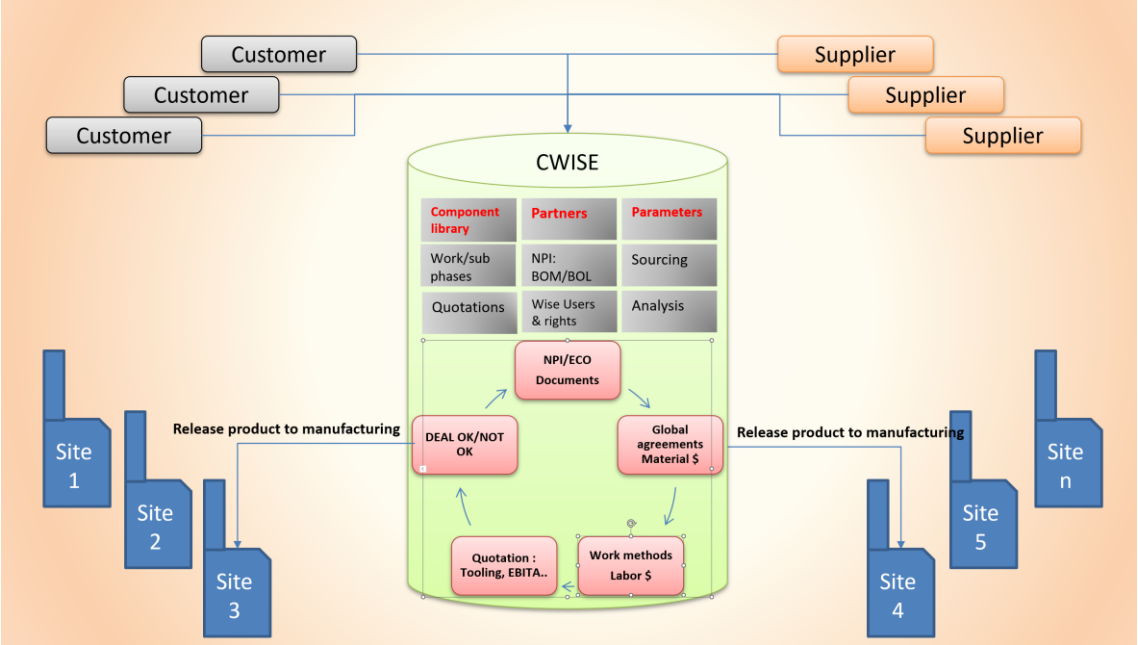
Työn ensisijaisena tavoitteena oli DSI-tiedoston raakadatan tallentaminen relaatiotietokantaan sektiokohtaisesti ja lisäksi kahden eri DSI:n vertailu käyttöliittymässä. Työ on osa laajempaa asiakasprojektia, jonka tavoitteina on luoda ja täydentää nimiketietoja toiminnanohjausjärjestelmään siihen luetuista johdinsarjoja kuvaavista tiedostoista ja siirtää muita valmistukseen liittyviä toimintoja paikallisista ohjelmistoista keskitettyyn CWise-järjestelmään. Luettavia tiedostoja on muutamaa eri tyyppiä.

2 TIETOPERUSTA

Luvussa esitellään Wisetimen toimittamat CWise- ja LAD-järjestelmät sekä DSI-tiedostomuoto. DSI-tiedostoista luotuja nimiketietoja tullaan käyttämään muissakin kuin tässä esitellyissä järjestelmissä, mutta nekin saavat ne CWisestä. LAD esitellään, koska sillä voidaan havainnollistaa, kuinka tietoja käytetään.

2.1 CWise

CWise eli Central Wise on keskitetty tiedonhallintajärjestelmä useiden eri tehtaiden Wise-järjestelmien tietojen vaihtamiseen ja hallintaan (kuva 1). CWisestä voidaan seurata tehtaiden suoriutumista maailmanlaajuisesti ja luoda niille yhteisiä nimiketietoja. Esimerkiksi tehtaille yhteisten työvaiheiden ja nimiketietojen perusteella voidaan verrata työvoimakuluja ja muita kustannuksia eri maissa ja tehdä globaalia hankintaa. (3.)



KUVA 1. Keskitetyn toiminnanohjauksen idea. Tuote lähetään valmistettavaksi maailmanlaajuisen kustannuskilpailutuksen perusteella. (3.)

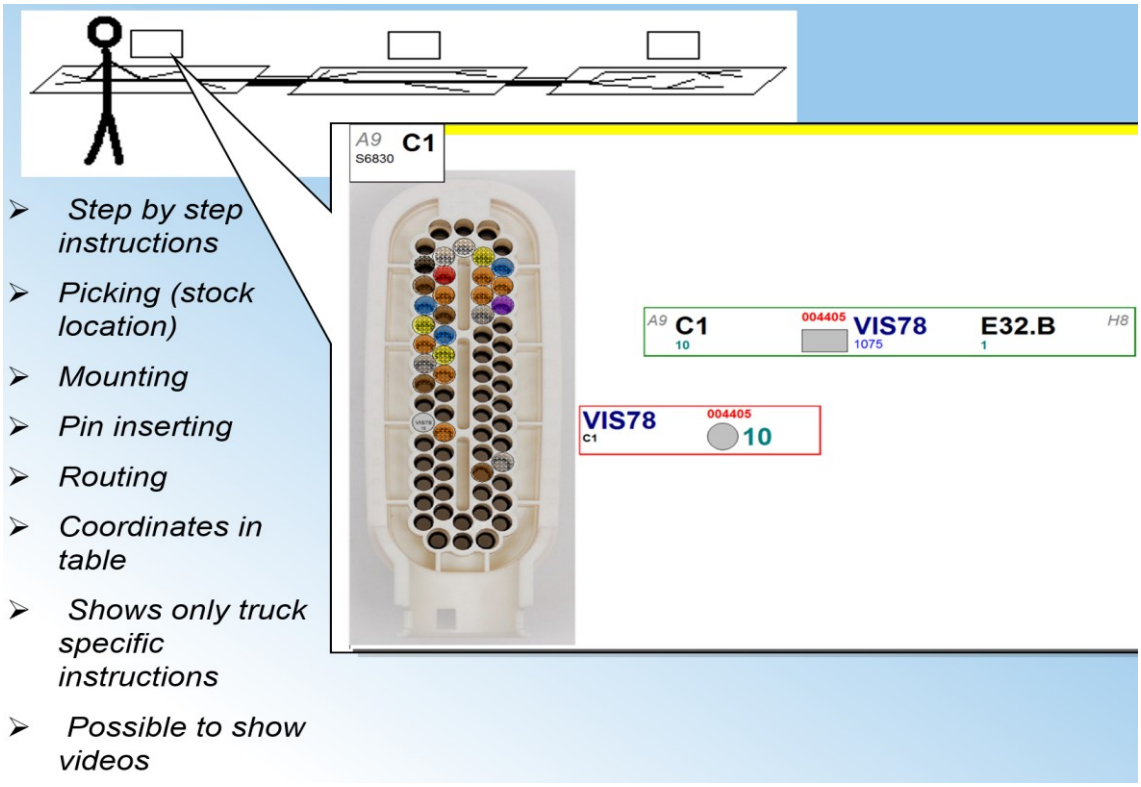
CWisen käyttöliittymä (kuva 2) on tehty Delphi-ohjelmankehitysympäristössä ja -kielellä. Käyttöliittymä lähettää tietokantakyselyjä ja -proseduurikutsuja Direct Oracle Access -komponenttien avulla.

CWisen-käyttöliittymä koostuu useista lomake- ja luetteloikkunoista. Luettelo näyttää jonkin aihealueen tietueet, joita voidaan määrittää ja päivittää lomakkeilla.

KUVA 2. CWisen päämenu. Luetut DSI-tiedostot löytyvät Drawing import -ikkunasta.

2.2 LAD

Line Assembly Developer (LAD) on järjestelmä johdinsarjojen loppukokoonpanoon ja erilaisten puolivalmisteiden valmistuksen ohjeistukseen ja ohjaukseen. Järjestelmä hallitsee useita tuotantolinjoja. LADiin määritetään työvaiheet jokaiselle työasemalle tuotantolinjalla (kuva 3). LADilla voidaan ohjata sekä synkro- että erätuotantoa. Erätuotannossa valmistetaan fyysisesti samanlaisia tuotekappaleita ja synkro-tuotannossa valmistetaan asiakkaan ominaisuuksiltaan yksilöimiä tuotteita. LAD on tuotantoa konkreettisesti ohjaava tietojärjestelmä, joka saa nimiketiedot toiminnanohjausjärjestelmältä ja on osa toiminnanohjauskokonaisuutta. Myös LAD on tehty Delphillä. (4.)



KUVA 3. Line Assembly Developerin toimintaperiaate kiteytettynä (5)

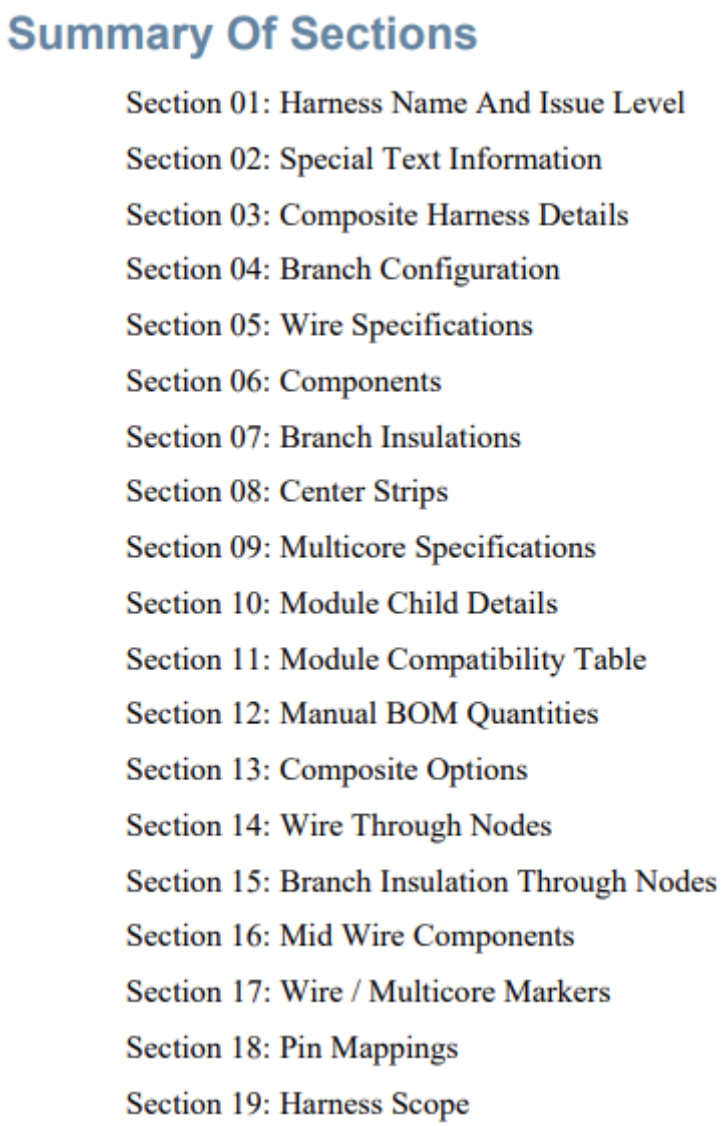
2.3 DSI-tiedostomuoto

DSI-tiedostot ovat ASCII-teksitiedostoja eli ne eivät sisällä ASCII-merkistöön kuulumattomia merkkejä, kuten ääkkösiä. Tiedosto koostuu useista sektioista, jotka sisältävät sektion oman määritelmän mukaisia tietueita. Tietueissa on useita kenttiä, joille on määritetty formaatti, kuten ”30 characters”, ”DD/MM/YY” tai ”999999.999999”. Tiedostossa on kolme erityismerkkiä: kommenttimerkki, sektioerotin ja kenttäerotin. Määrittelydokumentin mukaan erityismerkkeinä voidaan käyttää mitä tahansa ASCII-merkkiä, kunhan sitä ei käytetä tiedostossa mihinkään muuhun tarkoitukseen. CWiseen luettavat tiedostot käyttävät oletuseritysmerkkejä (taulukko 1). (6.)

TAULUKKO 1. DSI-tiedostomuodon oletuserityismerkit (6)

MERKKI	KUVAUS
!	Kommenttimerkki. Aloittaa kommentin, joka päättyy rivinvaihtoon. DSI-syötetiedostoja käsittelevät ohjelmat jättävät kommenttimerkillä alkavat rivit huomiotta.
%	Sektioerotin. Sektioerottimella tunnistetaan mihin sektio päättyy ja eri sektio alkaa. Sektiomerkkiä ei käytetä ensimmäisen sektion aloittamiseen. Tiedosto päätetään sektioerottimella.
:	Kenttäerotin. Erottaa kentät toisistaan eli kenttä päättyy aina kenttäerottimeen.

Opinnäytetyö tehtiin Capital Harness v12 DSI File Structure -määrittelydokumenttin mukaisesti. Dokumenttia on viimeksi päivitetty joulukuussa 2011 (6). CWiseen luettavat DSI:t on tehty version 7 määrittelyn mukaisesti, joka on julkaistu vuonna 2001 (7). Uudemmissa versioissa on lisätty sektioita ja kenttiä tai kasvatettu tietoformaattien kokoa. Esimerkiksi sektio 19 on lisätty versiossa 12. Sektiomäärittelyssä on merkitty kunkin kentän kohdalla, missä DSI-spesifikaatiossa se on lisätty tiedostomuotoon. Sektioiden tulee olla oikeassa järjestyksessä (kuva 4) ja niiden erotinmerkit tulee lisätä tiedostoon, vaikka niitä ei käytettäisi jossain johdinsarjassa.



KUVA 4. DSI-tiedoston sektiot (6)

Kuten sektiotkin, kentät ovat positionaalisia eli ensimmäinen kenttäerotin päättää ensimmäisen kentän, mikä ilmaisee seuraavan kentän alkamiskohdan. Tietueella pitäisi olla niin monta kenttäerotinta kuin sektion määrittelyssä on kenttiä (kuva 5). Sama pätee myös sektioerottimien ja sektioiden lukumäärän kohdalla. Tietotyypin kuvauksessa on voitu määrittää oletusarvo hakasuluissa, esimerkiksi loogiselle tietoformaatille [no]. Oletusarvot pitää huomioida vain DSI-tiedostoja tuottavissa ohjelmistoissa.

Section 01: Harness Name And Issue Level

This section of the input file only requires one record.

#	DSI	Field	Type	Format	Description
1	1	Harness Name 1	Text	30 characters	First harness part no. (customer harness of exporting system)
2	1	Harness Issue 1	Text	30 characters	First harness issue level.
3	1	Harness Date 1	Date	DD/MM/YYYY	First harness drawing release date
4	1	File Version	Number	99	DSI file version number (currently 12)
5	4	Customer Name	Text	30 characters	Customer name
6	4	Harness Name 2	Text	30 characters	Second harness part no. (internal harness of exporting system)
⋮					
38	4	Datum Route	Text	40 characters	Initial start route of branch configuration
39	5	Module Type	Logical		Is this harness a module? [no]
40	6	Alpha Code	Text	30 characters	Harness alpha code, only set in Designer from 3.0a and above.
41	7	Use pitch tables	Logical	yes / no	Should pitch tables be used for in-house twisted pair length calculation? [no]

KUVA 5. Esimerkki sektion 1 kenttien määrittelystä. Esimerkiksi 41. kenttä on lisätty DSI:hin versiossa 7. (6.)

```
!Bundle Drawing Name: 1234556 Bundle Drawing File: 1234556.txt
!Section 1: Harness Name And Issue Level
1234556:16:30/03/2004:7::::::::::::::A1::::::::::WIRES CAB LOWER::::::::no:A21.X1::yes:::
%Section 2: Special Text Information
!It is not supported currently.
%Section 3: Composite Details
1234556::30/03/2004::::::::::::::::::::::::::::::::::::001:LHD:
%Section 4: Branch Configuration
G45_1a::x0y0:splice::x-98y236:255:::018::::::::::
G45_2b::x1718y-143:CLO::x1783y-143:65:::024::::::::::
%Section 5: Wire Specifications
%Section 6: Components
%Section 7: Branch Insulations
%Section 8: Center Strips
!It is not supported currently.
%Section 9: Multicore Specifications
%Section 10: Module Child
%Section 11: Module Compatibility Table
!It is not supported currently.
%Section 12: Manual BOM Quantities
!It is not supported currently.
%Section 13: Composite Options
!It is not supported currently.
%Section 14: Wire Through Nodes
!It is not supported currently.
%Section 15: Branch Insulation Through Nodes
!It is not supported currently.
%Section 16: Mid Wire Components
!It is not supported currently.
%Section 17: Wire / Multicore Markers
!It is not supported currently.
%Section 18: Pin Mappings
!It is not supported currently.
%Section 19: Harness Scope
!It is not supported currently.
%End of File Marker
```

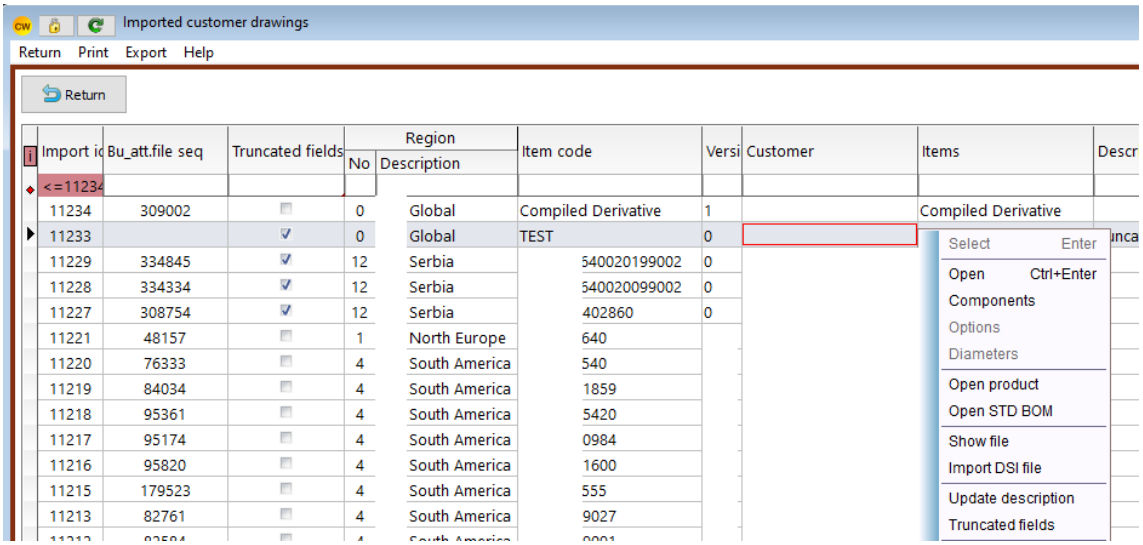
KUVA 6. Tekaistu, mutta spesifikaation mukainen DSI-tiedosto. Kuvassa sektioilla 1 ja 3 on yksi tietue ja sektiolla 4 kaksi tietuetta.

3 KEHITTÄMISTEHTÄVÄN KUVAUS

Kehittämistyöstä kerrotaan toteuttamisjärjestyksen vastaisesti alkaen käyttöliittymästä. Sen jälkeen kerrotaan DSI-tiedoston lukemisesta tietokantaan ja tuontityökalun testauksesta. Opinnäytetyön tulokset ovat nähtävissä tämän luvun kuvista. Lisäksi kerrotaan alkuperäisten tavoitteiden ulkopuolisista kehityksistä. Viimeisessä luvussa esitetään ohjelman suoritusajan mittaus.

3.1 Tiedostojen tuonti ja vertailu käyttöliittymässä

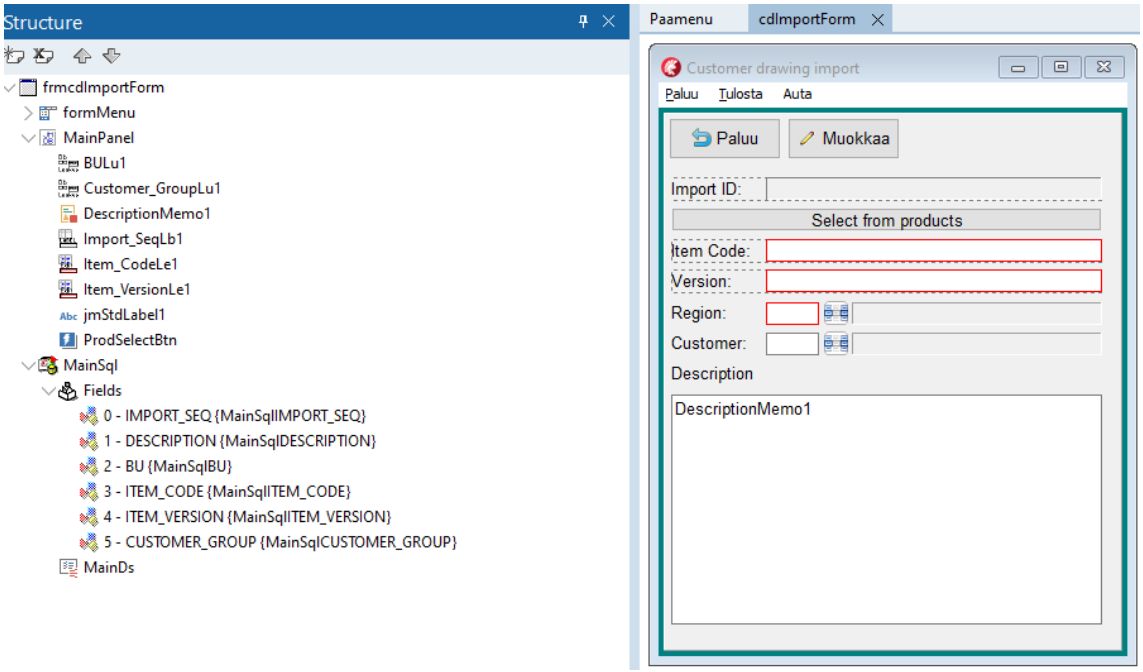
CWisen päämenuun lisättiin Drawing import -painike, josta aukeaa luettelonäkymä DSI-tiedostoista, jotka on tuotu järjestelmään. Luettelo näyttää tuonnit otsaketasolla, johon tallennetaan muun muassa lopputuotteen koodi ja versio. Otsaketaulusta on tietokantanäkymä eli tallennettu kysely, jossa taulun tietojen lisäksi haetaan tietoja käyttäjäystävällisemmin tekstinä numerotunnisteen sijaan ja kerätään DSI:stä löydetyt johdinsarjat pilkulla erotetuksi listaksi Oraclen LISTAGG-funktiolla.



KUVA 7. Luettelo CWise-järjestelmään luetuista DSI-tiedostoista

Luettelon ponnahdusvalikosta voidaan tuoda uusia tiedostoja järjestelmään tuontilomakkeella (kuva 8). Lomakkeen hyväksymisen jälkeen järjestelmä kysyy tiedostoja, joita yritetään jäsentää sektiotauluihin. Jos lukeminen epäonnistuu, käyttäjälle näytetään virheviesti ja tietokantatransaktio

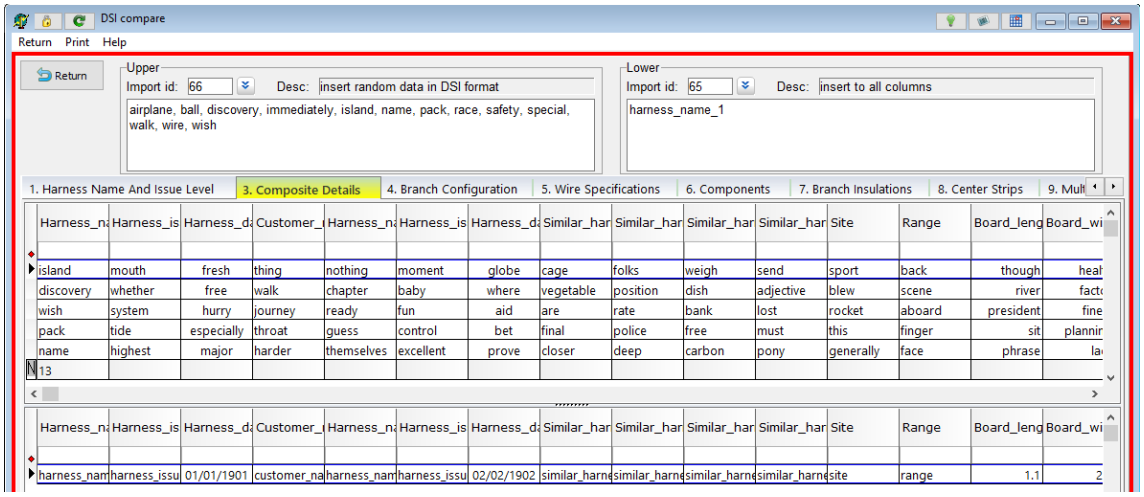
peruutetaan. Transaktio on joukko komentoja, jotka toteutetaan tai perutaan yhdessä. Myös CWise-järjestelmässä valmiiksi olevia liitetiedostoja voidaan lukea DSI-sektiotauluihin käyttöliittymän liitetiedostorekisteristä.



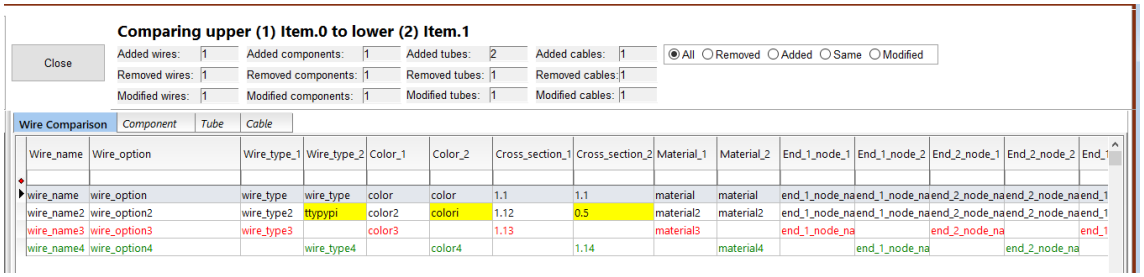
KUVA 8. Kehitysvaiheen DSI-tiedoston tuontilomake. Punaisella reunustetut tiedot ovat pakollisia.

Tuontiluettelon ponnahdusvalikosta voidaan myös avata DSI-tiedosto vertailuun (kuva 9). Raakavertailuikkunassa jokaiselle sektiolle on oma välilehti. Välilehti on näkyvissä käyttäjälle, jos sektiossa on vähintään yksi tietue. Välilehdissä on kaksi päällekkäistä tietoruudukkoa, joihin voi valita minkä tahansa järjestelmään luetun DSI:n. Vertailun apuna voidaan käyttää tietoruudukon funktioita, kuten tietueiden summaa, lajittelujärjestelyä ja sarakesuodatuksia eli tietokantakyselyn tulosten rajoittamista hakuehdon mukaisesti.

Vertailuikkunasta voidaan avata myös vertailutyökalu, joka tutkii tiedostojen eroja muutaman avainkentän avulla (kuva 10). Avainkentät yksilöivät tietueet molemmissa DSI-tuonneissa, joiden muita kenttiä verrataan lisäämällä ne uuteen tauluun Oraclen MERGE-lauseella. MERGEllä voidaan yhdistää datan manipulointikomentoja (DML), kuten lisäys ja päivitys, useista lähteistä yhdeksi komennoksi tapauksille, jossa avainkentät täsmäävät tai eivät täsmää (8).



KUVA 9. DSI-tiedostojen raakavertailuikkuna. Luettelon yläosassa on listattu tiedostosta löytyneet johdinsarjojen tuotekoodit pilkulla erotettuna.



KUVA 10. DSI-tiedostojen vertailutyökalu. Kuvassa kenttiä Wire_name ja Wire_option on käytetty avaimena johdinten vertailussa.

3.2 Lukeminen tietokantaan

CWisen data ja palvelinlogiikka on Oraclen relaatiotietokannassa. Palvelinlogiikka on toteutettu PL/SQL-kielelllä, joka on proseduraalinen laajennos SQL:ään eli se lisää muun muassa aliohjelmat, silmukat ja ehtolausekkeet.

DSI-tiedostoihin liittyvät proseduurit ovat koottu muutamaan tietokantapakettiin. Paketteihin laitetaan toisiin liittyviä aliohjelmia ja muita tietokantaobjekteja. Kun tuontilomakkeen otsaketiedot on tallennettu, kutsutaan paketissa olevaa ohjelmaa, joka jäsentää tiedoston sektiotauluihin. Jokaiseen sektiotauluun tietueeseen tallennetaan otsaketaulun pääavain. Toisin sanoen otsaketaulun ja sektiotaulujen välillä on yhden suhde moneen -yhteys.

Sektiotaulussa oleva otsaketaulun pääavain on indeksoitu. Indeksi on tauluun tallennettu hakemistorakenne, jolla nopeutetaan haettavan tiedon löytymistä. Indeksiin tallennetaan indeksoidut kentät (tai kenttä) ja taulun rivien sijainti, joista kentät löytyvät (9). Sijaintien tietäminen nopeuttaa taulun skannausta tietokantakyselyissä, jos oletetaan, että indeksin arvo ei esiinny liian tiheään suhteessa tietueiden lukumäärään.

Ennen tiedoston lukemista tarkistetaan muutamalla ehdolla, vaikuttaisiko tiedosto olevan DSI-formaatin mukainen (kuva 11). Tässä vaiheessa tällä pyritään välttämään turhan tiedon tallentamista ja huomauttamaan käyttäjälle epäsopivasta tiedostosta. Myöhemmin tätä tullaan käyttämään tiedostojen massaluvussa, jossa voidaan lukea useampaa tuettua tiedostotyyppiä kerralla.

FileIsDsi

```
Function FileIsDsi(ppImportSeq in pls_integer, ppBuFileSeq in pls_integer default null) return boolean
```

FUNCTION FILEISDSI

Description: returns true if input file has:
comment lines before fist data record into section 1
first data record has more than 38 fields (41 as per v12 spec)
more than 17 section markers

Params
ppImporSeq in integer
ppBuFileSeq in integer default null (Bu_Attachments_File_pk)

Returns
boolean

Called from
parse_dsi_into_tables

KUVA 11. DSI-tiedostoformaatin tunnistavan aliohjelman dokumentointi

DSI luetaan sektiotauluihin käsittelemällä tekstiä rivi kerrallaan silmukassa (kuva 12). Rivin päättymiskohta tunnistetaan rivinvaihtomerkistä. Rivinpituus eli rivin merkkien lukumäärä lasketaan etsimällä seuraavan rivinvaihtomerkin sijainti ja vähentämällä siitä kertyneen siirtymän. Siirtymä kertoo mistä rivinvaihtomerkin etsintä aloitetaan. Silmukan lopussa siirtymää kasvatetaan rivin ja rivinvaihtomerkin pituudella.

DSI:n ensimmäinen ei-kommenttirivi on tietue ensimmäiseen sektiotauluun, koska sektioerotinta ei käytetä ensimmäisen sektion merkkaukseen vaan se on kommenttina. Kun sektioerotinmerkki havaitaan, sektiolaskuria kasvatetaan. Käsiteltävänä oleva tekstirivi jäsennetään oikeaan sektiotauluun CASE-valintarakenteessa. Tapauksessa, jossa sektiolaskuri on 3, rakennetaan tietue kolmanteen sektiotauluun ja niin edelleen.

```
-- Loop CLOB row by row
while pvOffset <= pvLengthTotal
loop
  pvRowNr := pvRowNr + 1;
  -- Search position of next line ending and calculate line length
  pvLineLength := instr(pvClob, CHR(10), pvOffset) - pvOffset;
  if pvLineLength < 0 then
    -- If CHR(10) not found starting at offset (at last line), instr() returns 0
    -- => pvLineLength = -pvOffset < 0
    -- pvOffset has accumulated to include length of all lines except last
    pvLineLength := pvLengthTotal - pvOffset + 1;
  end if;

  -- Extract current line
  pvLineData := substr(pvClob, pvOffset, pvLineLength);
  pvLineFirstChar := substr(pvLineData, 1, 1);

  if pvLineFirstChar = ppCommentDelim or pvLineData is null then
    -- Ignore comment and empty rows
    null;
  elsif pvLineFirstChar = ppSectionDelim then
    if upper(pvLineData) not like '\%END%' escape '\' then
      -- count actual sections, not End of File Markers
      pvSectionCounter := pvSectionCounter + 1;
    end if;
  else
  -- Build record into section table N = pvSectionCounter
```

KUVA 12. DSI:n käsittely silmukassa. Tiedosto on character large object -tyyppisessä pvClob-muuttujassa. (10.)

Sektiotaulun tietue rakennetaan kenttä kerrallaan. Kenttään haetaan arvo DSI:n rivistä kenttäerottimen esiintyvyyden perusteella (kuva 13). Sektioiden 3 ja 19 viimeisissä kentissä voi määrittelyn mukaisesti olla useampi sektioerottimella erotettu arvo, joten niihin luetaan loppurivi viimeistä kenttää edeltävän sektioerottimen jälkeen. Muihin sektioihin lisättiin kuusi ylimääräistä kenttää: viisi yhden arvon sisältävää ja viimeinen kenttä loppuriviä varten. Nämä kentät lisättiin, koska virheellisissä DSI-tiedostoissa voi olla kenttien siirtymää, esimerkiksi kolmanteen kenttään tarkoitettu data onkin neljännessä kentässä ja vastaavasti muissa, jolloin viimeinen kenttä jäisi lukematta. Uudemmat DSI-spesifikaatiot voivat myös lisätä uusia kenttiä sektioihin. Tärkeintä on saada kaikki data tietokantaan. Sitä voidaan myöhemmin järjestää uudelleen.

```
  if ppIsDate then
    -- Some DSIs' date formats like: 12/9/2020 12:00:00 AM ; 08.01.2022 00:00:00,
should be: DD/MM/YYYY
    pvWrongDateFormat := FieldDataByColumnNumber(ppColNr)
      ||':'||FieldDataByColumnNumber(ppColNr+1)
      ||':'||FieldDataByColumnNumber(ppColNr+2);
    if REGEXP_LIKE(
      pvWrongDateFormat,
      '\d{1,2}[/.]\d{1,2}[/.]\d{4} \d{2}:\d{2}:\d{2}*'
    ) then
      pvDate := REGEXP_SUBSTR(pvWrongDateFormat, '\d{1,2}[/.]\d{1,2}[/.]\d{4}');
      pvDate := replace(pvDate, '.', '/');
      pvLineData := REGEXP_REPLACE(pvLineData, pvWrongDateFormat, pvDate, 1, 1);
    end if;
  end if;
  -- save field number for exception messages
  pvCurrentFieldNr := ppColNr;

  -- get ppInvoker's column size
  open ColumnLengthCur;
  fetch ColumnLengthCur into pvTruncateLen;
  close ColumnLengthCur;

  -- get data before field delimiter at occurrence ppColNr
  pvResult := REGEXP_SUBSTR(
    pvLineData,
    '[^'||pcFieldDelim||']*'||pcFieldDelim,
    1,
    ppColNr
  );
  -- Remove terminating field delim
  pvResult := RTRIM(pvResult, pcFieldDelim);

  if length(pvResult) > pvTruncateLen and pvTruncateLen is not null then
    RAISE value_error;
    -- Exception handler catches raised exception
    -- and truncates result to fit ppInvokers column size
  end if;

  return pvResult;
```

KUVA 13. Kentän arvon hakeminen kenttänumeron perusteella. Aluksi korjataan virheelliset päivämäärät. Data tarvittaessa katkaistaan mahtumaan taulun kenttään, ja tallennetaan katkaisemattomana erilliseen lokitauluun.

Jäsennysohjelmassa on itse määriteltyjä virheviestejä esimerkiksi tapauksille, jossa sektioita on enemmän kuin 19 tai taulun kenttään lisättävä data on epäsopivaa tietotyyppiä. Tietotyyppivirhe

nostetaan esimerkiksi, kun satamerkkiseen taulun kenttään yritetään laittaa pidempää merkkijonoa tai implisiittinen päivämäärän formaatin muuntaminen ei onnistu. Tietotyyppivirheen viestissä ilmaistaan virheen asiayhteys lisäämällä siihen kenttä- ja rivinumero, sektiolaskurin arvo ja data, joka aiheutti virheen (kuva 14). Ohjelman lopussa on poikkeuskäsittelijä, joka nappaa muut virheet ja nostaa käyttäjälle Oraclen virheviestin ja virheen jäljityksen eli miten ohjelmaa on suoritettu, jotta on päädytty virhetilanteeseen.

```
when 19 then
  pvSec19Row.Import_Seq := ppImportSeq;
  pvSec19Row.Harness_Name_1 := FieldDataByColumnNumber(
    1,
    'Harness_Name_1'
  );
  -- Fields 2..4
  pvSec19Row.Scope_Names := RowDataAfterFieldNr(5, 'Scope_Names');
  insert into cd_s19_harness_scope values pvSec19Row;
else
  pvErMsg := 'Counted more than 19 sections, count: '||pvSectionCounter;
  Raise_application_error(-20000, pvErMsg);
END CASE;

EXCEPTION
  when value_error then
  pvErMsg := 'Field: '||pvCurrentFieldNr
    ||' at section: '||pvSectionCounter
    ||' at row: '||pvRowNr
    ||' raised VALUE_ERROR';
  Raise_application_error(-20000, pvErMsg);
end;
```

KUVA 14. Jäsennysohjelman tietotyyppivirheen poikkeuskäsittely. Kuvasta näkee myös tietueen rakentamisen CASE-valintarakenteen viimeisessä tapauksessa.

DSI:n lukemisen yhteydessä muutaman sektion muistiinpanokenttä jäsennetään omiksi kentiksi. Muistiinpano kenttä voi sisältää useita tietoja, esimerkiksi ”length = 1, var = a + b”. Tällöin lengthin ja varin arvo tallennettaisiin omiin kenttiin. Nämä ovat valmistajakohtaisia tietoja, jotka eivät kuulu DSI-määrittelyyn.

Lisäksi sektiotauluista kaivetaan sarjakohtaiset tiedot omiin tauluihin. DSI-tiedostoon voidaan lisätä useita johdinsarjoja antamalla yksittäiselle sarjalle optiokoodin sektiossa 3. Composite details tai sektiossa 10. Module child details. Esimerkiksi tietyn johtimen tietueeseen on lisätty niiden sarjojen

optiokoodit, joihin johdin kuuluu. Muita sarjakohtaisia tietoja ovat haaran konfiguraatio, komponentit, eristeet ja kaapelit.

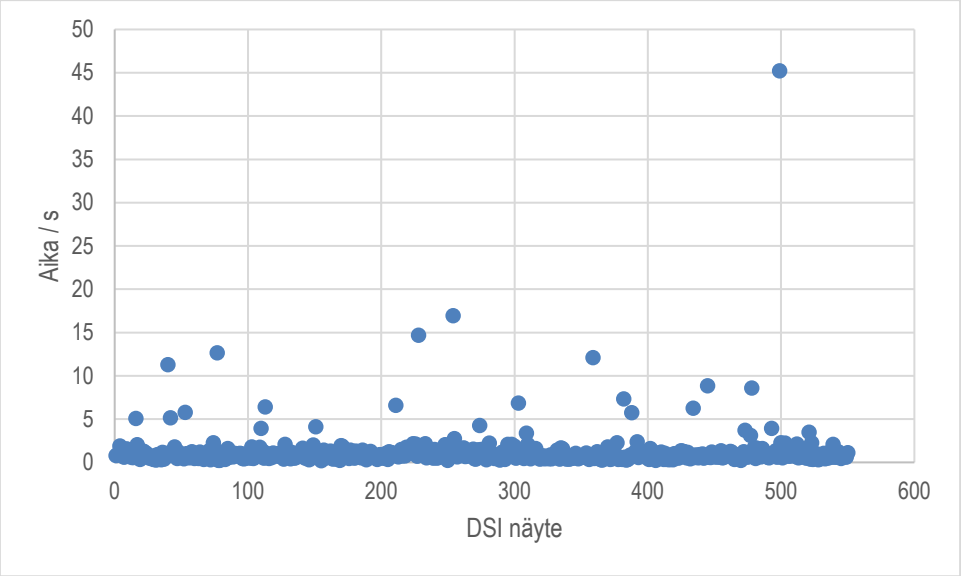
Lopulta kuorma-auton valmistajan komponenttikoodit tallennetaan sarjakohtaisesti erilliseen tauluun tarvittavan kappalemäärän kanssa. Tähän tauluun haetaan johdinsarjan valmistajan vastaavat tuotekoodit asiakkaan koodin perusteella. Taulun tietoja hyödyntäen kehitettiin käyttöliittymäominaisuus, jossa verrataan DSI-tiedostosta luettuja komponentteja CWisen lopputuoterakenteeseen. Lopputuoterakennetta voi päivittää, jos siitä puuttuu jokin DSI-tiedostosta luettu komponentti.

3.3 DSI-tuonnin testaus

Jäsennysohjelmaa testattiin kehitysvaiheessa itse tehdyllä DSI-tiedostolla, jossa jokaiseen määrittelyssä olevaan kenttään laitetaan dataa. Lisäksi muutamalta kuorma-autovalmistajalta valittiin kourallinen satunnaisia testitiedostoja. Kun mainitut saatiin luettua järjestelmään ilman ongelmia, aloitettiin lukeminen massana. Viime vuosina CWisen lopputuotteen liitetiedostoihin on tallennettu sen DSI-tiedosto. Näitä oli CWisen demoympäristössä yli kymmenentuhatta tiedostoa. Massaluvun yhteydessä virheitä aiheuttaneiden tiedostojen tunnisteet otettiin talteen ja lukemista jatkettiin. Virheitä aiheuttaneita tiedostoja, jotka olivat DSI-tiedostoja, oli noin kymmenen. Testitiedostojen sekaan tuli muitakin tiedostomuotoja.

Tuonnin testauksen tuloksena kenttätietoja alettiin tarvittaessa katkomaan sopimaan varchar2-tietotyyppiin eli enintään 4000 merkkiin (tavuun). Katkaisu tehdään kyseessä olevan kentän kokoon, joka on useimmiten 100 merkkiä. Katkaisematon data tallennetaan erillisen lokitaulun suureen tekstiobjekti-kenttään.

Ohjelman suoritusaikaa mitattiin jäsentämällä viitisen prosenttia aiemmin luetuista DSI-tiedoista uudelleen (kuva 15). Suoritusaika tallennettiin lokitauluun itsenäisessä transaktiossa ja jäsennyksen transaktion peruutettiin. Mittauksessa ei havaittu mitään uutta. Tiedostot, joissa on vain yksi johdinsarja, saadaan luettua nopeasti, muissa voi kestää kymmeniä sekunteja. Mittaustuloksista otettiin talteen tilastotietoa Oraclen agregaattifunktiolla, kuten STATS_MODElla (taulukko 2).



KUVA 15. Jäsennysohjelman suoritusaika

TAULUKKO 2. Ohjelman suoritusajan yhteenveto

Keskiarvo	*Mediaani*	*Moodi*	*Varianssi*	*Keskihajonta*
1,21	0,76	0,18	5,97	2,44

Opinnäytetyössä kehitetyt ominaisuudet ovat olleet asiakkaan testattavana, ja saadun palautteen perusteella esimerkiksi sallitaan myös .TEXT-päätteisten tiedostojen lukeminen ja pidennettiin tiedostopolun maksimipituutta.

3.4 Ongelmat ja ratkaisut

Yhdeksi ongelmaksi tuli erään DSI-tiedostoja tuottavan ohjelman tapa käyttää kaksoispistettä kellonajanerottimena. Tämä aiheutti kahden kentän siirtymää tietueissa, koska kaksoispistettä käytetään kenttäerottimena, jolloin päivämäärä tulkittiin päättyväksi ensimmäiseen kellonajanerottimeen. Ongelma ratkaistiin tutkimalla päivämääräkenttiä säännöllisillä lausekkeilla (kuva 13). Säännöllisellä lausekkeella tutkitaan merkkijonon vastaavuutta erilaisia sääntöjä ja malleja vasten (11). Päi-

vämäärän formaatin korjaamisen jälkeenkin saman ohjelman tuottamissa DSI-tiedostoissa on joissakin sektioissa siirtymää. Jokaisen sektion kaikki kentät testattiin niin, että niihin voi lisätä dataa määrittelydokumentin kenttänumeron perusteella. Täten siirtymä ei synny jäsennysohjelmasta.

Sektiotaulut tehtiin aluksi liian tarkasti määrittelyn mukaan. Päivämääräkenttien tietotyypiksi laitettiin DATE, liukuluvun merkitseviksi numeroiksi ja skaalaksi täsmälleen saman kuin spesifikaatiossa ja niin edelleen. Tämä aiheutti tietotyyppivirheitä, koska DSI-tiedostot eivät ole aina määrittelyn mukaisia. Nyt kaikki kentät luetaan merkkijonoina ja niiden merkkikoossa on tarpeeksi puskuria, sillä ne ovat usein 2–3 kertaa suurempia kuin spesifikaatiossa.

Lisäksi tekstitiedostot lähetettiin suoraan tekstiobjektina käyttöliittymästä. Tämä tuotti ongelmia, jos tiedoston ja tietokannan enkoodaukset eivät olleet samoja. Nyt tiedostot lähetetään binääriobjekteina, jonka tavujärjestysmerkistä tunnistetaan tekstin enkoodaus ja muunnetaan tekstiobjektiksi kannassa. Muunnokseen löytyi sopiva ohjelma avoimella MIT-käyttölisenssillä (12).

4 YHTEENVETO

Opinnäytetyölle asetetut tavoitteet saatiin toteutettua, omaa alkuperäistä tiukahkoa aikataulua lukuun ottamatta. CWise-järjestelmään saadaan luettua DSI-tiedostoja ja niitä voidaan verrata käyttöliittymässä. Työt näiden aiheiden ympärillä jatkuvat. Seuraavaksi kehitetään rajapinta tietojen siirtoon eri järjestelmiin, tuontiohjelmat muille tiedostotyypeille ja materiaalitarkastus luetuille asiakaskoodeille, missä etsitään tai luodaan vastaavat valmistajan koodit.

Opin työn aikana paremmin käyttämään Oraclen merkkijonofunktioita; lähes koko jäsennysohjelma perustuu niiden käyttöön. Lisäksi opin tekstinkäsittelyä eri ympäristöissä, kuten esimerkiksi Notepad++:ssa, Excelissä ja PL/SQL Developerissa, pitkien ja toistuvien komentojen rakentamisesta ja DSI-tiedostojen tutkimisesta. Työn jälkeen ymmärrän paremmin johdinsarjojen valmistuksen prosessia toiminnanohjauksen näkökulmasta, minkä parissa jatkan työskentelyä.

LÄHTEET

1. Wisetime 2016. Wisetime Oy. Hakupäivä 19.4.2022. https://www.wisetime.fi/Wisetime_Oy_www.pdf.
2. Hayes, Caroline 2020. Mentor changes its name to Siemens EDA. Electronics Weekly. Hakupäivä 25.4.2022. https://www.electronicsweekly.com/news/products/fpga-news/mentor-changes-name-siemens-eda-2020-12/.
3. Wisetime 2017. Wise ERP PowerPoint-esitys. Sisäinen lähde.
4. Wisetime 2022. Technical Document LAD. Sisäinen lähde.
5. Wisetime 2016. Instructor document. Sisäinen lähde.
6. Mentor Graphics 2011. Capital Harness v12 DSI File Structure. Sisäinen lähde.
7. Mentor Graphics 2001. DSI File Specification Version 7. Sisäinen lähde.
8. Oracle 2022. Oracle Database SQL Language Reference, 21c. Hakupäivä 26.7.2022. https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/MERGE.html#GUID-5692CCB7-24D9-4C0E-81A7-A22436DC968F.
9. Saxon, Chris 2017. How to Create and Use Indexes in Oracle Database. Hakupäivä 25.4.2022. https://blogs.oracle.com/sql/post/how-to-create-and-use-indexes-in-oracle-database.
10. Haack, Andy 2018. Reading clob line by line with pl/sql. Hakupäivä 29.3.2022. https://stackoverflow.com/a/37703019.
11. Linux.fi 2020. Säännöllinen lauseke. Hakupäivä 26.4.2022. https://www.linux.fi/wiki/S%C3%A4%C3%A4nn%C3%B6llinen_lauseke.
12. DidiSoft Inc. Blob2clob. Hakupäivä 25.7.2022. https://github.com/didisoft/blob2clob.