KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Mikko Kähkönen

KOKOONPANOLINJAN DATAN VISUALISOINNIN AUTOMATISOINTI

Opinnäytetyö
Maaliskuu 2020

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Maaliskuu 2020
Tietotekniikan koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä
Mikko Kähkönen

Nimeke
Kokoonpanolinjan datan visualisoinnin automatisointi

Toimeksiantaja
Phillips-Medisize Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli kehittää tietokanta sekä sovellus kokoonpanolinjan tiedon varastointiin ja visualisointiin. Sovellusta käytettäisiin kokoonpanolinjan prosessin kehittämisen ja saavutetun tason ylläpitämisen työkaluna. Tiedon esittämisen lisäksi sovelluksen avulla kerättäisiin tietoa kokoonpanolinjan aikarakenteesta. Järjestelmä otettaisiin myöhemmin käyttöön myös muilla toimeksiantajan kokoonpanolinjoilla.

Järjestelmän tietokanta luotiin Microsoftin SQL Server 2016:lle. Sovellus kehitettiin C#-ohjelmointikielellä Windows Presentation Foundation -kirjastoa ja MVVM-mallia käyttäen. Suuren tietomäärän esittämiseksi kehitettiin ratkaisu, jossa kokoonpanoprosessin toimintojen suorituskykyä kuvataan värimatriiseilla.

Vaikka opinnäytetyöprojektin suunnittelu oli vajaata ja aikataulu venyi, tuloksena saatiin toimiva järjestelmä, joka on monistettavissa muille kokoonpanolinjoille konfiguraatiomuutoksilla. Yhdenmukainen tietokantamäärittely kokoonpanolinjojen kesken on antanut pohjan uusien linjojen tiedonkeruun suunnittelussa.

Kieli
suomi

Sivuja 27
Liitteet 0
Liitesivumäärä 0

Asiasanat
C#, MVVM, Entity Framework, SQL Server, Käyttöliittymä, Tuotantolinja, Kokoonpanolinja, Valvomo, Visualisointi

THESIS
March 2020
Degree Programme in Information Technology
Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author
Mikko Kähkönen

Title
Automated assembly line data visualization

Commissioned by
Phillips-Medisize Oy

Abstract

The aim of this thesis was to develop a database and an application for storing and visualizing of information gathered from an assembly line. The application would be used as a tool for improving and maintaining performance of the assembly process. In addition to data visualization, time loss data would be collected with the application. The system would later be introduced to other similar assembly lines.

The database was created for Microsoft SQL Server 2016. The application was developed with C# using Windows Presentation Foundation framework and MVVM-pattern. To present a large amount of data, a solution was developed where the performance of assembly process steps are displayed as heatmaps.

As a result of this thesis, a functional system was developed despite the project not finishing on time due to incomplete planning. The resulting system can be deployed to other assembly lines with configurational changes. The unified database structure across assembly lines has been used as a base for data collection of new assembly lines.

Language
Finnish

Pages 27
Appendices 0
Pages of Appendices 0

Keywords
C#, MVVM, Entity Framework, SQL Server, User interface, Manufacturing line, Assembly line, Control dashboard, Visualization

Sisältö

Käsitteet

Entity Framework 6	Microsoftin kehittämä olio-relaatiomallintaja .NET-ympäristöön [1].
HMI	Human-Machine Interface, ohjelmoitavan logiikan käyttöliittymä.
KNL	Käytettävyys, nopeus, laatu. Katso OEE.
Laskennallinen sarake	Relaatiotietokannan taulukon sarake, jonka arvo lasketaan mahdollisesti muiden sarakkeiden arvoja käyttäen.
MVVM	Sovellusarkkitehtuurimalli, jossa sovellus jaetaan kolmeen luokkaan: malliin (Model), näkymään (View) ja näkymämalliin (ViewModel).
OEE	Overall equipment effectiveness. Tuotantokoneiden kokonaistehokkuus. [2.]
Syöttölaitehäiriö	Häiriö tuotantokoneessa, jonka vuoksi asetettavaa osaa ei ole saatavilla.
Tallennettu proseduuri	Tietokantaan tallennettu kysely.
Valvomo	Tuotannon prosessin mittareiden käyttöliittymä.
WPF	Windows Presentation Foundation. Microsoftin kehittämä käyttöliittymärajapinta Windowsille. [3.]

1 Johdanto

Massatuotantoympäristössä prosessien jatkuvaa kehittämistä vaaditaan kilpailukyvyn ylläpitämiseksi. Automaattisella kokoonpanolinjalla tuotantomäärät ovat suuria ja prosessikokonaisuus monimutkainen, joten ihmisen on vaikea huomata mahdollisia ongelma- tai kehityskohtia ilman prosessista kerättyä dataa. Monimutkaisesta prosessikokonaisuudesta saa valtavasti dataa, joten arvo saadaan vasta kun data on jalostettu käytettäväksi tiedoksi. Vanhaa tietoa voidaan käyttää työkaluna prosessin kehittämiseen, mutta todellinen prosessin tehokkuuden ylläpito vaatii reagointia prosessin ei-haluttuihin muutoksiin. Parhaiten reagoinnin mahdollistaa ajan tasalla oleva tieto.

Opinnäytetyön toimeksiantajana toimii Kontiolahtelainen Molex-konserniin kuuluva Phillips-Medisize Oy, joka valmistaa pääasiassa muovisia lääkeannostelulaitteita. Tuotteet kootaan toimeksiantajan itsensä ruiskuvalamalla valmistetuista komponenteista. Tämän opinnäytetyön tarkoitus oli kehittää valvomokokonaisuus, joka olisi monistettavissa toimeksiantajan muille kokoonpanolinjoille. Kokoonpanolinjan datan visualisoinnin automatisoinniksi olisi kehitettävä tietokanta ja valvomosovellus. Valvomosovellusta ajettaisiin kokoonpanolinjan läheisyydessä. Tiedon varastoinnin ja esittäminen lisäksi oli kerättävä tietoa kokoonpanolinjan ajankäytöstä, jotta voitiin laskea tarkka OEE-mittari.

Koska toimeksiantajan kokemuksesta hukkatavara on ollut suurin yksittäinen tehokkuuteen negatiivisesti vaikuttava tekijä, keskittyy kehitettävä valvomosovellus olennaisesti laadun valvonnan kontrollien seurantaan. Komponenttien saatavuuden maksimointi, eli syöttölaitehäiriöiden minimointi, on kokoonpanolinjan läpimenoajan kannalta tärkein tekijä, johon tuotannon työntekijät voivat vaikuttaa. Syöttölaitehäiriöiden valvonta on valvomosovelluksen toinen pääkohde.

2 Alkutilanne

Toimeksiantajalla on useita eri tuotteita valmistavia kokoonpanolinjoja. Useimpien kokoonpanolinjojen tarkastellaan linjan omilta HMI-näytöiltä, jotka ovat usein monimutkaisia ja kankeita käyttää. Joillakin kokoonpanolinjoilla on olemassa oleva tiedonkeruu toimeksiantajan omistamiin tietokantoihin, mutta tietoja ei käytetä tehokkaasti hyväksi. Tarvetta uudelle järjestelmälle lisäsi toimeksiantajan onnistunut hukkaosien määrän tiputukseen pyrkinyt vuoden kestänyt projekti ja uuden palvelinympäristön hankinta. Koska kohdekokoonpanolinjalla ei ollut riittävää valvomosovellusta saavutetun tason säilyttämisen tukemiseksi, mutta linjalla oli olemassa oleva tiedonkeruujärjestelmä ja tarvittavat verkkoyhteydet, valittiin se projektin pääasialliseksi kohteeksi.

Kohdekokoonpanolinja kokoaa insuliinikyniä pääosin samalla tehtaalla ruiskuvaletuista muovikomponenteista ja sen maksimikapasiteetti on 13 200 kynää tunnissa. Kokoonpanolinja koostuu soluissa olevista työasemista. Koottavat tuotteet kulkevat työasemien läpi telakoilla, joissa on useita pesiä. Työntekijät ovat tottuneet kutsumaan tuotantolinjojen eri osia näillä termeillä ja niiden englanninkielisillä vastineilla: cell, station ja nest. Kokoonpanolinjalla on syöttölaitteiden ja työasemien lisäksi myös laadunvalvonta-asemia, jotka mittaavat työn laatua ja tarvittaessa hylkäävät tuotteen. Laatuasemista puhuttaessa käytetään pesän sijaan termiä kontrolli eli control.

Kokoonpanolinjalta kerättiin ennestään tietoa kahteen tietokantaan, jotka sijaitsivat eri palvelimilla. Toiseen tietokantaan kerättiin linjan käyntitiedot, eli onko linja käynnissä vai ei, ja toiseen tiedot hyvien osien, huonojen osien ja syöttölaitehäiriöiden osalta. Tuotantotiedot ovat aina sidottu soluun, asemaan ja pesään tai kontrolliin. Koska kokoonpanolinjan pitkien seisokkien syitä ei kerätty, täydellistä KNL-aikarakennetta ei voitu saatavilla olevien tietojen puitteissa rakentaa. Vaikka tältä kokoonpanolinjalta ei kerätty hälytystietoja, tultaisiin niitä keräämään tulevaisuudessa muilta linjoilta. Linjan hälytystiedot sisältävät tiedon miksi kokoonpanolinja on pysähdyksissä linjan sisäisen logiikan mukaan.

3 Tavoitteet

Tavoitteena oli kehittää tietokanta kohdekokoonpanolinjalle. Tietokannan taulurakenne otettaisiin tulevaisuudessa käyttöön myös muiden kokoonpanolinjojen tiedonvarastoinnissa. Yhtenäinen tietokantarakenne yksinkertaistaisi tiedon käsittelyä ja selkeyttäisi uusien kokoonpanolinjojen tiedonkeruun vaatimuksia. Tähän mennessä uusia kokoonpanolinjoja hankkiessa tiedonkeruu on rakennettu ulkopuolisen linjan rakentajan käytäntöjen mukaan, eikä toimeksiantajalla ole ollut siihen liittyviä vaatimuksia. Tietokantamäärittelyjen olisi siis oltava monistettavissa muille kokoonpanolinjoille.

Projektin päätavoitteena oli kehittää valvomotyyppinen sovellus tietokannan tiedon visualisointiin, joka mahdollistaisi tiedon käytön kokoonpanolinjan toimintaan liittyvässä päätöksenteossa ja jota käytettäisiin pitkien linjaseisokkien syiden keräämiseen. Käyttöliittymän tulisi olla intuitiivinen ja tukea kosketusnäyttökomentoja, sillä sen pääasiallinen käyttö tapahtuisi suurella kosketusnäyttötelevisiolla. Sovelluksen olisi oltava arvokas päätöksenteon työkaluna nopeasti vilkaistaessa ja tietoa tarkemmin tarkastellessa, sillä sovelluksella olisi useita eri käyttäjäryhmiä erilaisin tarpein. Sovellusta ajettaisiin kellon ympäri kokoonpanolinjan läheisyydessä. Sovelluksen olisi oltava itsenäinen kokonaisuus, sillä tuotantoympäristöjen verkkoyhteydet ovat usein suljettuja ja projektin onnistuessa sovellusta käytettäisiin useita vuosia. Myös sovellus otettaisiin käyttöön tulevaisuudessa muilla kokoonpanolinjoilla ja sovellus tulisi olla konfiguroitavissa eri tietokantayhteyksille.

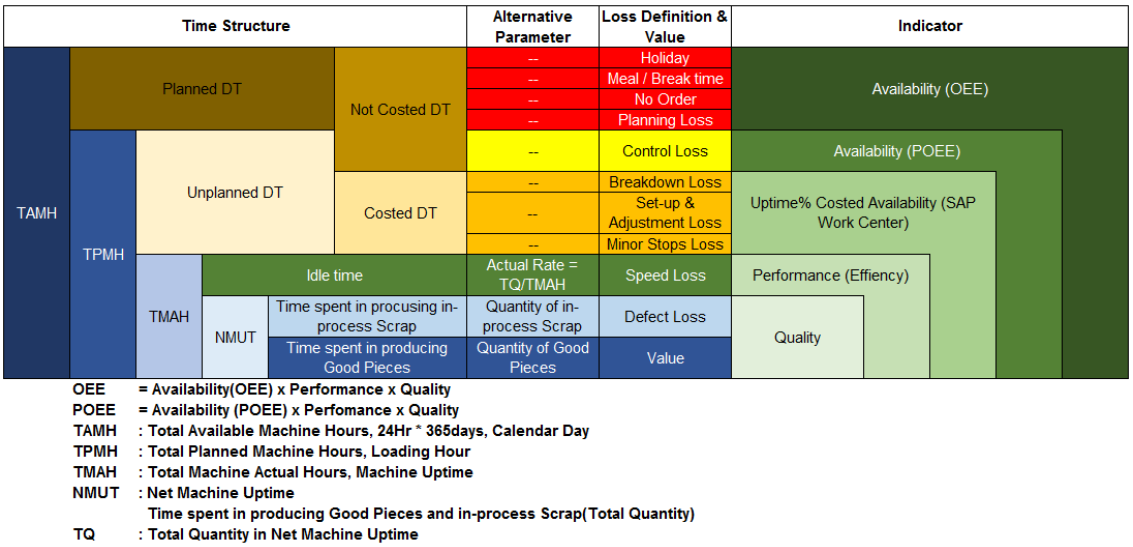
4 Vaatimukset

Tietokannan oli oltava nykyisten parhaiden käytäntöjen mukainen, normalisoitu ja monistettavissa asetusmuutoksilla samankaltaisille kokoonpanolinjoille. Koska tietokantapalvelimen resurssien käyttöä suositeltiin valvomosovelluksen tietokoneen resurssien sijaan, tietokantaan täytyisi luoda tietoa käsittelevät näkymät ja tallennetut proseduurit. Näkymiä ja proseduureja käytettäisiin mahdollisesti myös muissa yhteyksissä. Toimeksiantaja määritteli tietokantaohjelmistoksi Microsoft SQL Server 2016:n. Tietokannan oli oltava yhteensopiva olemassa olevan tiedonkeruun kanssa ja se oli oltava myöhemmin monistettavissa muille kokoonpanolinjoille kohtuullisilla muutoksilla.

Valvomo-ohjelmiston täytyisi olla itsenäinen Windows-sovellus ja sen käyttö tapahtuisi 55-tuumaisella kosketusnäytöllä. Sovelluksen käyttöjärjestelmänä toimisi pitkän ajan tuen Microsoft Windows 10 LTSC. Koska sovellusta ajettaisiin kellon ympäri, virhetilanteet tulisi käsitellä siten, että sovellus jatkaisi odotettua toimintaansa ilman käyttäjän komentoja esimerkiksi verkkovirheen jälkeen.

Kokoonpanolinjan toiminnan tiedot tulisi olla selkeästi visualisoitu ja tärkeimmät tunnusluvut helposti saatavilla sovelluksen päänäytöllä. Käyttäjän täytyisi nähdä sovelluksesta, jos seurattavan linja vaatii korjaavia toimintoja. Tämänhetkisen tilanteen lisäksi sovelluksella täytyisi olla mahdollista tutkia kokoonpanolinjan toiminnan historiaa ja tutkia tarkemmin kerättyä tietoa.

Sovellus käsittelisi ainakin hukkatavara-, syötelaitehäiriö- ja käyntitietoja. Lisäksi sovelluksesta olisi oltava saatavilla kuvassa 1 kuvatun KNL-aikarakenteen yleismittarit. Vaatimuksissa ei tarkemmin määritelty, kuinka tieto näytettäisiin ja millainen käyttöliittymän asettelu olisi. Määrittelyvaiheessa oli selvää, että vaatimukset uudelleen määriteltäisiin useasti projektin aikana näiden osa-alueiden osalta.



Kuva 1. OEE-aikarakenne.

Jos kokoonpanolinja olisi pysähdyksissä yli 5 minuuttia, käyttöliittymä pyytäisi käyttäjää valitsemaan pysähdyksen syyn ja tallentaisi valitun syyn tietokantaan. Tietokannassa tulisi olla mahdollista muokata valittavia syitä ja niiden metatietoja.

Sovelluksen kehityksen ja testikäytön ei tulisi häiritä olemassa olevan järjestelmän toimintaa. Tiedonkeruuta simuloitaisiin siirtämällä ja mukauttamalla olemassa olevien tietokantojen tiedot uuteen tietokantaan, kunnes lopullisessa käyttöönotossa tiedonkeruu tapahtuisi suoraan uuteen tietokantaan.

5 Valinnat

Koska sovelluksen käyttöjärjestelmäksi oli toimeksiantajan puolesta määritelty pitkän ajan tuen Windows 10 -versio, päädyttiin käyttämään Windows Presentation Foundation (WPF) kirjastoa sovelluksen graafisena rajapintana ja ohjelmointikielenä C#:a. Universal Windows Platform (UWP) ja Windows Forms olivat vaihtoehtoiset rajapinnat, mutta UWP hylättiin pitkän ajan tuen version riittämättömän komponenttitarjonnan vuoksi. UWP:llä SQL-yhteyksien muodostaminen on mahdollista vain Windows 10 Fall Creators Updaten jälkeisillä versioilla [4]. Vaikka Windows Forms tarjosi tarvittavan käytettävyyden, valittiin modernimpi WPF graafiseksi rajapinnaksi.

WPF:n natiivissa Code behind -sovellusmallissa ohjelman logiikka sijaitsee käyttöliittymän näkymien taustakoodissa. Tämä mahdollistaa yksinkertaisten sovellusten nopean kehittämisen, mutta taustakoodissa sijaitsevan logiikan ylläpidettävyys on heikkoa. Vaihtoehtona Code Behind -sovellusmallille on malli-näkymä-näkymämalli (eng. Model-View-ViewModel, MVVM), jossa sovelluksen käyttämä tietomalli, sisäinen logiikka ja käyttäjän näkemä näkymä erotetaan selkeästi toisistaan. MVVM-mallissa sovellus on ylläpidettävämpi ja testattavampi, sillä sovelluksen logiikka sijaitsee itsenäisissä näkymämalleissa [5]. Koska kehitettävä sovellus tulisi olemaan laaja monimutkainen kokonaisuus ja koska sovelluksen vaatimusmäärittelyt eivät olleet yksiselitteiset, lähdekoodin ylläpidettävyyden arvo korostui entisestään ja MVVM valittiin sovelluksen sovellusmalliksi. Koska WPF tukee MVVM-mallia hyvin, päätettiin MVVM toteuttaa ilman kolmannen osapuolen ohjelmistokomponenttikirjastoja.

Sovelluksessa tiedon visualisointiin valittiin LiveCharts-visualisointikirjasto. Kirjasto vastasi vaatimuksia riittävällä tasolla ja tuki MVVM-sovellusmallia. Beto Rodriquezin kehittämä LiveCharts erottui muista ehdokkaista edullisuudellaan ja osittain MIT-lisensoidulla avoimella lähdekoodillaan [6].

Koska kaikki alustat ovat Microsoftin tuotteita, valittiin Microsoftin Visual Studio sovelluksen ohjelmointiympäristöksi. Vanhojen tietokantojen tiedon integroinnissa uuteen tietokantaan käytettiin Visual Studion SQL Server Data Tools -työkalua. SQL Serverin tietokannan kehitykseen käytettiin Microsoftin SQL Server Management Studiota.

6 Toteutus

6.1 Tietokanta

Tietokantaan luotiin taulut kokoonpanolinjan perustiedoille. Perustietotaulujen tiedot tulee täyttää tietokannan käyttöönoton yhteydessä. Perustiedot sisältävät linjan tuotantotietojen lisäksi kontrolli-, syöttölaite-, tuotantosolu-, hälytys-,

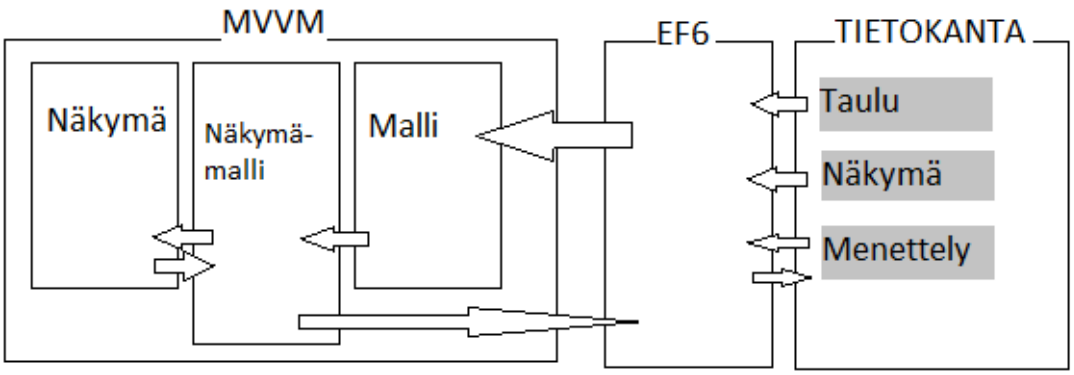
pysähdys- ja tuotantosolumäärittelyt. Perustietomäärittelyt mahdollistavat tietokannan monistamisen myöhemmin muille kokoonpanolinjoille.

Kokoonpanolinjan seuratut tapahtumat, kuten kontrollin havainto huonosta osasta, kirjataan tapahtumatauluihin perustietoihin viitattuna. Laskuritietoja, kuten solun läpi menneet hyvät osat, päivitetään minuutin välein minuuttitauluihin. Kaikesta tiedosta valmistetaan tuntikohtaista koostetietoa tuntitauluihin, joista tietokannan tiedot pääasiassa käsitellään eteenpäin. Koska tämän tietokannan tarkoitus ei ole validoida prosessia myöhempää tarkastelua varten, vaan kerätä tietoa prosessin toiminnasta, yksittäisen osan kulkua linjalla tai tarkempia jäljitettävyystietoja ei tallenneta. Tietokantaan kehitettiin taulu- ja proseduurirakenne prosessin osa-alueiden oletusarvojen laskentaan, tallennukseen ja vertailuun. Tietokanta sisältää myös näkymät ja laskentaproseduurit kokoonpanolinjan prosessin oleellisten KPI-mittareiden osalta.

6.2 Sovelluksen arkkitehtuuri

Koska valmista MVVM-arkkitehtuurikehystä päätettiin olla hyödyntämättä, implementoitiin MVVM-arkkitehtuurin toimintaan tarvittavat luokat Rachel Limin Simple MVVM Example esimerkin mukaan. Esimerkissä määritellään RelayCommand-luokka komentojen välitystä varten ja ObservableObject-luokka yksinkertaistamaan sen perivien luokkien ominaisuuksien päivitysten ilmoitusten hallinta [7].

Tämän sovelluksen MVVM-arkkitehtuurin malli on pääosin Entity Framework 6-oliorelaatiomallintajan generoima oliomalli sovelluksen tietokannasta. Oliomalli sisältää tietokannan taulujen ja proseduurien palautustaulujen tietueita vastaavien .NET-luokkien lisäksi tiedonsiirron toiminnallisuuden. Ohjelman logiikka sijaitsee näkymämalleissa, joiden tehtävä on käsitellä mallin tiedot ja käyttäjän komennot. Näkymään kiinnitetään näkymämallin ominaisuudet, jotka sisältävät näytettävät tiedot. Näkymät ovat XAML-tiedostoja, joissa määritellään käyttöliittymän asettelu ja tiedon esitysmuoto.

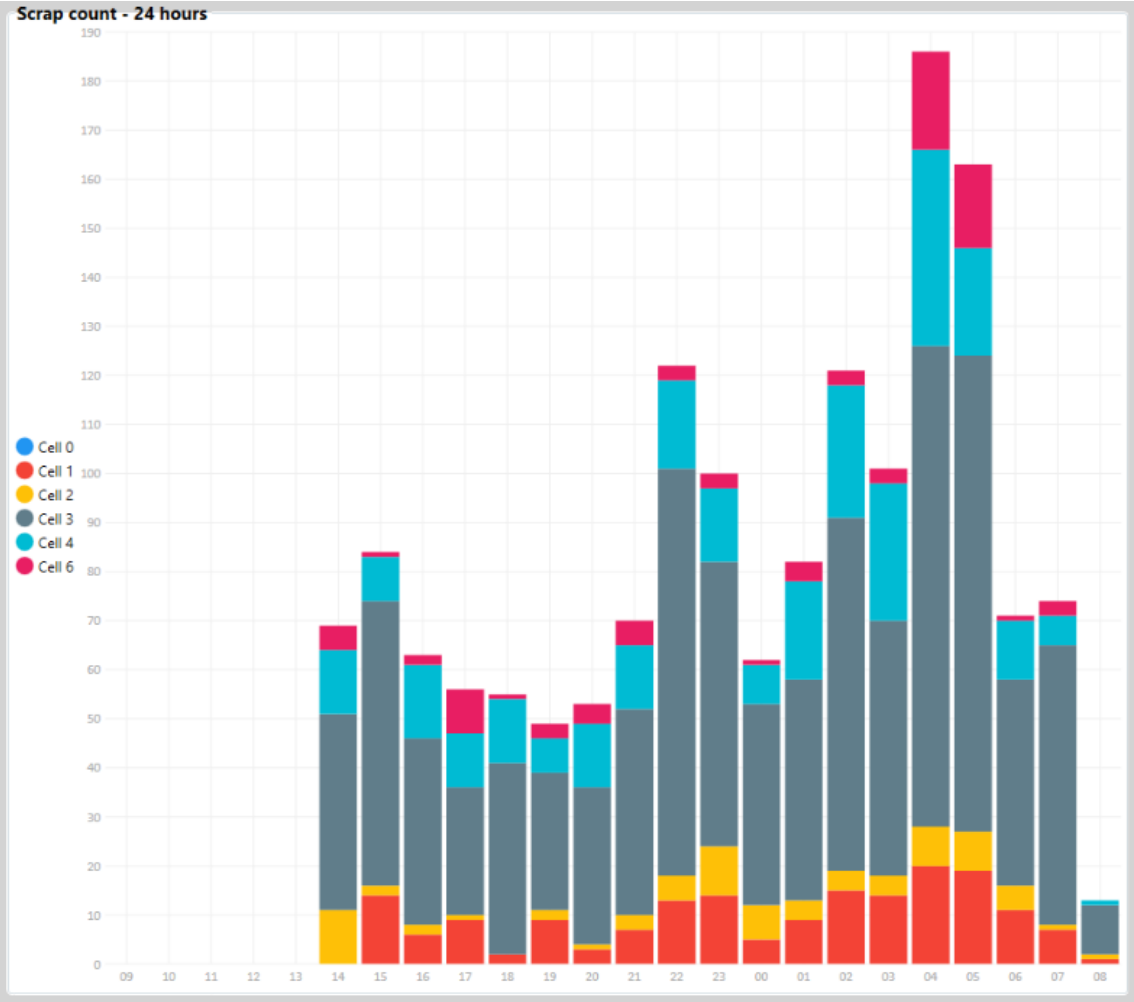


Kuva 2. Järjestelmän tiedonkulku.

Sovelluksessa käytetään LiveCharts-visualisointikirjastoa tiedon visualisoinnin työkaluna. Näkymämalli käsittelee mallin tiedon LiveCharts-kirjastolle yhteensopivaksi (kuva 2) ja näkymässä tieto sidotaan LiveCharts-komponenttiin. LiveCharts-view-komponentit ovat normaaleja WPF:n näkymäkomponentteja, joten tiedonsidonta ja käyttäjän komentojen käsittely voidaan toteuttaa MVVM-periaatteiden mukaan.

6.3 Tiedon visualisointi

Tietoa havainnollistetaan muun muassa taulukoilla, pylväskaavioilla, viivakaavioilla ja ympyrädiagrammeilla. Tiedon visualisoinnissa käytettiin usein olemassa olevien Excel-taulukoiden kaltaisia ratkaisuja. Käyttäjät olivat tottuneet lukemaan tietoa esimerkiksi kuvan 3 mukaisesta pylväskaaviosta.



Kuva 3. Kokoonpanolinjan tuotettu hukka / tunti / kokoonpanolinjan solu.

Nämä perinteiset tiedon visualisointimenetelmät ovat käytännöllisiä, kun näytettävien tietokokonaisuuksien määrä ei kasva liian suureksi. Ongelmaksi laatukontrollien ja syöttölaitteiden tiedon visualisoinnissa muodostui tarvittavien kaavioiden lukumäärä. Oli ratkaistava, kuinka yli sadan toisistaan riippumattoman kokonaisuuden tiedot voidaan tiivistää samaan näkymään, kuitenkin heikentämättä näkymäkokonaisuuden selkeyttä. Pareto-kaavioiden käyttäminen ei olisi ratkaissut ongelmaa täysin, sillä mahdollisesti tärkeää tietoa olisi jäänyt näyttämättä.

Koska toiminnon virhetaajuutta voidaan ajatella virheen todennäköisyydeksi, toiminnon normaaliuus voidaan kuvata P-valvontakorttia käyttäen. Koska otoskoot eivät ole vakioita, tiedon visualisoinnissa käytettiin hyväksi otoskoon suhteen standardoitua P-valvontakorttia. Näin saadaan standardoitu matriisi kokoonpanolinjan toimintojen normaaliudesta. Tämän matriisin pohjalta tieto voidaan visualisoida värikarttoina.

Otoksen standardoitu P-arvo Z lasketaan kaavalla:

$$Z_i = \frac{\hat{p}_i - \bar{p}}{\sqrt{\frac{\bar{p}(1-\bar{p})}{n_i}}}$$

, jossa $\hat{p}_i$ on tutkittavan otoksen virheprosentti, $\bar{p}$ on pitkän ajan laskettu todennäkyisyys virheelle ja n_i on otoskoko. Jos Z on pienempi kuin 0, otoksen virheprosentti on pienempi kuin oletettu pitkän ajan virheprosentti ja jos Z on suurempi kuin 0, otoksessa on enemmän virheitä, kuin normaalissa tilanteessa. Yli ylävalvontarajan 3 oleva Z arvo tarkoittaa otoksen virheprosentin olevan merkittävästi huonompi kuin normaali taso.

Jotta Z-arvojen laskenta tapahtuisi mahdollisimman nopeasti ja koska toimeksiantaja toivoi mahdollisuutta määritellä pitkän ajan virhekeskiarvot valitulta aikaväliltä, tietokantaan tallennetaan käsiteltävien tietojen pitkien ajan virhekeskiarvot. Kaaviossa YY näytetään, kuinka kaava XX johdettiin muotoon, josta kantaan voitiin laskennan nopeuttamiseksi laskea ennalta myös toinen termi.

$$Z_i = \frac{\hat{p}_i - \bar{p}}{\sqrt{\frac{\bar{p}(1-\bar{p})}{n_i}}}$$

$$\Leftrightarrow\ Z_i = (\hat{p}_i - \bar{p}) \frac{1}{\sqrt{\bar{p}(1-\bar{p})}\ \frac{1}{\sqrt{n_i}}}$$

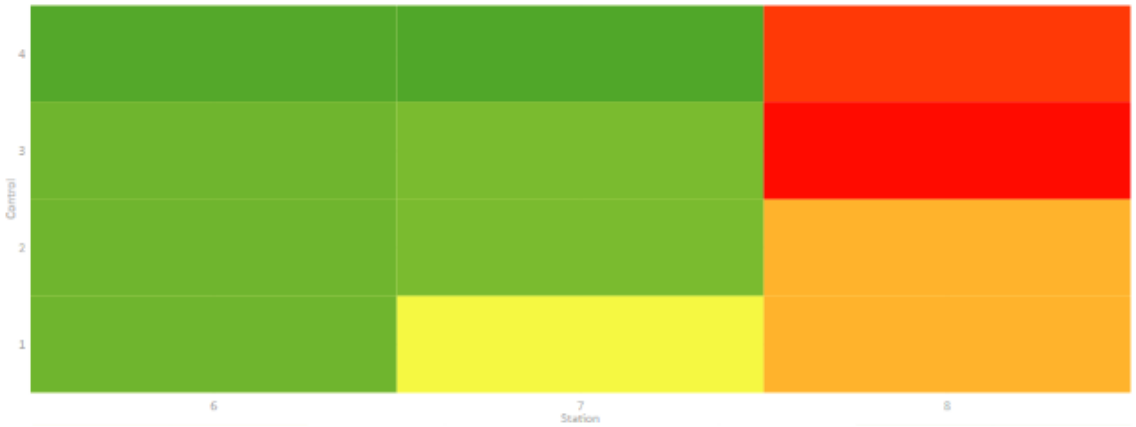
$$\Leftrightarrow Z_i = \ (\hat{p}_i - \bar{p})\ \frac{1}{\sqrt{\bar{p}(1-\bar{p})}}\ \sqrt{n_i}$$

Tietokantaan määriteltiin johdetun kaavan toisen termin mukainen laskennallinen sarake PBarFactor. Taulukko 1 kuvaa kuinka tietokannan Z-arvojen laskennan arvot tallennetaan tietokantaan. Tietokanta palauttaa tarvittaessa PBar-taulun ja otoksen tiedoista lasketut Z-arvon.

Taulukko 1. Syöttölaitteen tallennetut PBar-taulun arvot.

Sarake	**Selite**
ID	Rivitunniste
Feeder_ID	Syöttölaitteen ID
PBar	Pitkän ajan hylkäyskes-kiarvo
PBarFactor	Laskennallinen sarake $\frac{1}{\sqrt{\mathrm{PBar}(1-\mathrm{PBar})}}$
…	…

Toiminnan normaaliutta kuvaavassa lämpökartassa Z-arvot muunnetaan väri-koodeiksi. Jos toiminta on normaalia tai oletettua parempaa, eli Z-arvo on liki-määrin tai pienempi kuin 0, lämpökartan sen toiminnan solu on vihreä. Keltainen ja oranssi väri kuvaa normaalia suurempaa virhetaajuutta, joka kuitenkin pysyy kontrollirajojen sisällä. Jos Z on suurempi kuin kontrolliraja 3, eli jos virheitä kirja-taan epänormaalin paljon, toimintaa kuvaava väri on punainen.



Kuva 4. Kokoonpanolinjan solun laatukontrollien lämpökartta.

Kuva 4 kuvaa kokoonpanolinjan erään solun laatukontrollien toiminnan lämpö-karttaa. Kuvatun solun asema 8 hylkää valitulla aikavälillä tuotteita selvästi ole-tettua enemmän ja varsinkin kontrollien 3 ja 4 hylkäysprosentit ovat merkittävän suuret suhteessa kontrollien normaaleihin tasoihin.

6.4 Tiedonkulku

Tässä luvussa esitellään prosessista kerätyn tiedon käsittely tietokannasta sovelluksen käyttöliittymään. Tarkempaan tarkasteluun otetaan syöttölaiteiden toimintaa kuvaavat kahdeksan edellisen tunnin automaattisesti päivittyvät lämpökartat esimerkin vuoksi.

Kun linjalla ilmenee syöttölaitehäiriö, syöttölaitteen sensori ilmoittaa häiriön ja häiriö kirjataan kantaan syöttölaitehäiriötauluun *FeederError* (taulukko 2) häiriössä olevan syöttölaitteen tunnuksella (taulukko 3). Häiriön loputtua syöttölaitehäiriötauluun päivitetään virhetilan lopetusaika ja kesto.

Taulukko 2. Feeder-taulu.

Sarake	**Selite**
ID	Syöttölaitteen tunnus
Cell_ID	Syöttölaitteen solun tunniste
Station	Syöttölaitteen asema
Nest	Syöttölaitteen pesä
PartDescription	Syöttölaitteen osan kuvaus
Sensor	Syöttölaitteen sensori

Taulukko 3. FeederError-taulu.

Sarake	**Selite**
ID	Syöttölaitevirheen ID
Feeder_ID	Syöttölaitteen ID
StartedAt	Virhetilan aloitusaika
EndedAt	Virhetilan lopetusaika
Duration	Virheen laskettu kesto

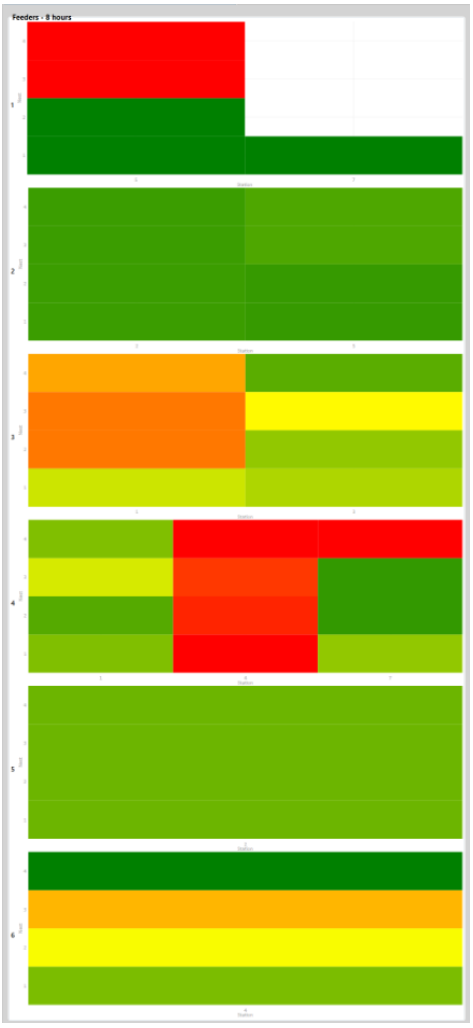
Tallennettu proseduuri GetFeederSummary vastaanottaa parametreinä tarkisteltavan aikavälin ja palauttaa taulukon 4 mukaisen syöttölaitteiden toimintaa kuvaavan taulun.

Taulukko 4. GetFeederSummary – tallennetun proseduurin palauttama taulu.

Sarake	**Selite**
ID	Syöttölaitteen ID
Cell_ID	Syöttölaitteen solun tunniste
Station	Syöttölaitteen asema
Nest	Syöttölaitteen pesä
PartDescription	Syöttölaitteen osan kuvaus
ErrorTimePct	Prosenttia ajasta häiriössä
MinutesInErrors	Minuutit häiriössä
ErrorCount	Häiriöiden määrä
TotalOutput	Linjan kokonaistuotanto
PBar	Syöttölaitteen pitkän ajan keskiarvo-häiriö suhteessa kokonaistuotantoon. Todennäköisyys häiriöön.
PBarFactor	Pitkän ajan keskiarvosta laskettu vakio otoskoon muutoksien käsittelyyn
Z	Standardoitu P-valvontakortti

Kehitetty tietokanta on mallinnettu sovellukseen Entity Framework 6 -oliorelaatiomallintajaa käyttäen. Tietokantamalli sisältää viittaukset ja tarvittavat kutsut tietokannan tauluihin, näkymiin ja tallennettuihin proseduureihin. Mallin tallennettua proseduuria vastaava funktio palauttaa tietokannan palauttamaa taulua vastaavan oliolistan. Oliolista voidaan tallentaa sellaisenaan näkymämallin ominaisuuteen. Näkymään on määritelty näkymäelementit, joihin näkymämallin ominaisuudet sidotaan. Kaikki sovelluksen näkymämallit periytyvät *ObservableObject*-yläluokasta, johon on määritelty luokkien ominaisuuksien muutosten ilmoitus näkymille. Kun sidottujen ominaisuuksien tietoja muokataan, päivittyy näkymä automaattisesti.

Kuvassa 5 esitellään sovelluksen päänäkymän syöttölaitteiden lämpökartat. Päänäkymän näkymämalli johtaa *GetFeederSummary*-tallennetun proseduurin palauttamasta oliolistasta listan *HeatMapModel*-olioita (kuva 7), yhden jokaista kokoonpanolinjan solua kohden.

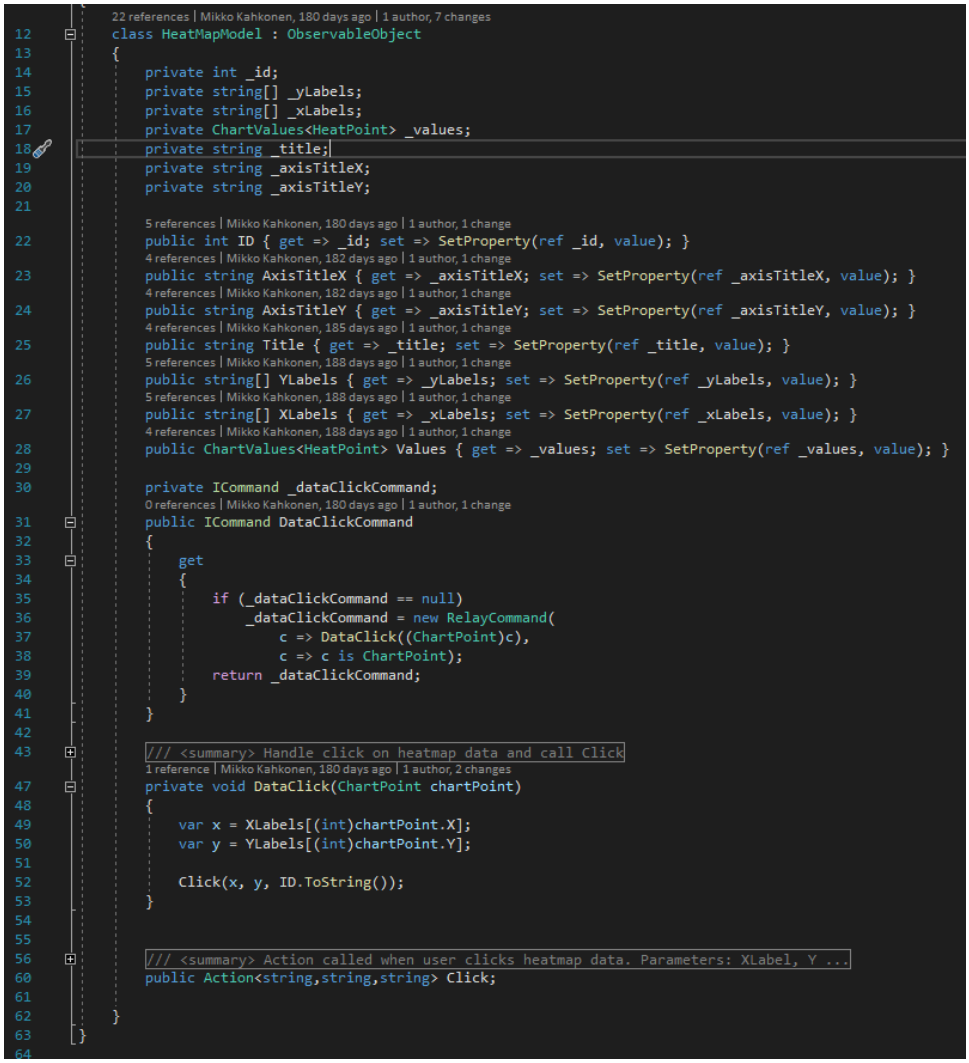


Kuva 5. Solujen syöttölaitteiden lämpökartat.

Koska *HeatMapModel*-luokka toimii myös lämpökartta-näkymäelementin näkymämallina, luotu lista voidaan sitoa suoraan näkymään. Kuvassa 6 esitetyn XAML-määrittelyn mukaan WPF luo *HeatMap*-näkymän ja sitoo tiedon jokaista listan oliota kohden.

```
<ItemsControl ItemsSource="{Binding HeatMapList}"
              Grid.Column="1"
              Name="HeatMapColumn">
    <ItemsControl.ItemsPanel>
        <ItemsPanelTemplate>
            <UniformGrid Rows="{Binding HeatMapList.Count}" />
        </ItemsPanelTemplate>
    </ItemsControl.ItemsPanel>
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            <charts:HeatMap DataContext="{Binding }" />
        </DataTemplate>
    </ItemsControl.ItemTemplate>
</ItemsControl>
```

Kuva 6. Lämpökarttalistan XAML-määrittely näkymässä.



```
22 references | Mikko Kahkonen, 180 days ago | 1 author, 7 changes
class HeatMapModel : ObservableObject
{
    private int _id;
    private string[] _yLabels;
    private string[] _xLabels;
    private ChartValues<HeatPoint> _values;
    private string _title;
    private string _axisTitleX;
    private string _axisTitleY;

    5 references | Mikko Kahkonen, 180 days ago | 1 author, 1 change
    public int ID { get => _id; set => SetProperty(ref _id, value); }
    4 references | Mikko Kahkonen, 182 days ago | 1 author, 1 change
    public string AxisTitleX { get => _axisTitleX; set => SetProperty(ref _axisTitleX, value); }
    4 references | Mikko Kahkonen, 182 days ago | 1 author, 1 change
    public string AxisTitleY { get => _axisTitleY; set => SetProperty(ref _axisTitleY, value); }
    4 references | Mikko Kahkonen, 185 days ago | 1 author, 1 change
    public string Title { get => _title; set => SetProperty(ref _title, value); }
    5 references | Mikko Kahkonen, 188 days ago | 1 author, 1 change
    public string[] YLabels { get => _yLabels; set => SetProperty(ref _yLabels, value); }
    5 references | Mikko Kahkonen, 188 days ago | 1 author, 1 change
    public string[] XLabels { get => _xLabels; set => SetProperty(ref _xLabels, value); }
    4 references | Mikko Kahkonen, 188 days ago | 1 author, 1 change
    public ChartValues<HeatPoint> Values { get => _values; set => SetProperty(ref _values, value); }

    private ICommand _dataClickCommand;
    0 references | Mikko Kahkonen, 180 days ago | 1 author, 1 change
    public ICommand DataClickCommand
    {
        get
        {
            if (_dataClickCommand == null)
                _dataClickCommand = new RelayCommand(
                    c => DataClick((ChartPoint)c),
                    c => c is ChartPoint);
            return _dataClickCommand;
        }
    }

    /// <summary> Handle click on heatmap data and call Click
    1 reference | Mikko Kahkonen, 180 days ago | 1 author, 2 changes
    private void DataClick(ChartPoint chartPoint)
    {
        var x = XLabels[(int)chartPoint.X];
        var y = YLabels[(int)chartPoint.Y];

        Click(x, y, ID.ToString());
    }


    /// <summary> Action called when user clicks heatmap data. Parameters: XLabel, Y ...
    public Action<string,string,string> Click;

}
}
```

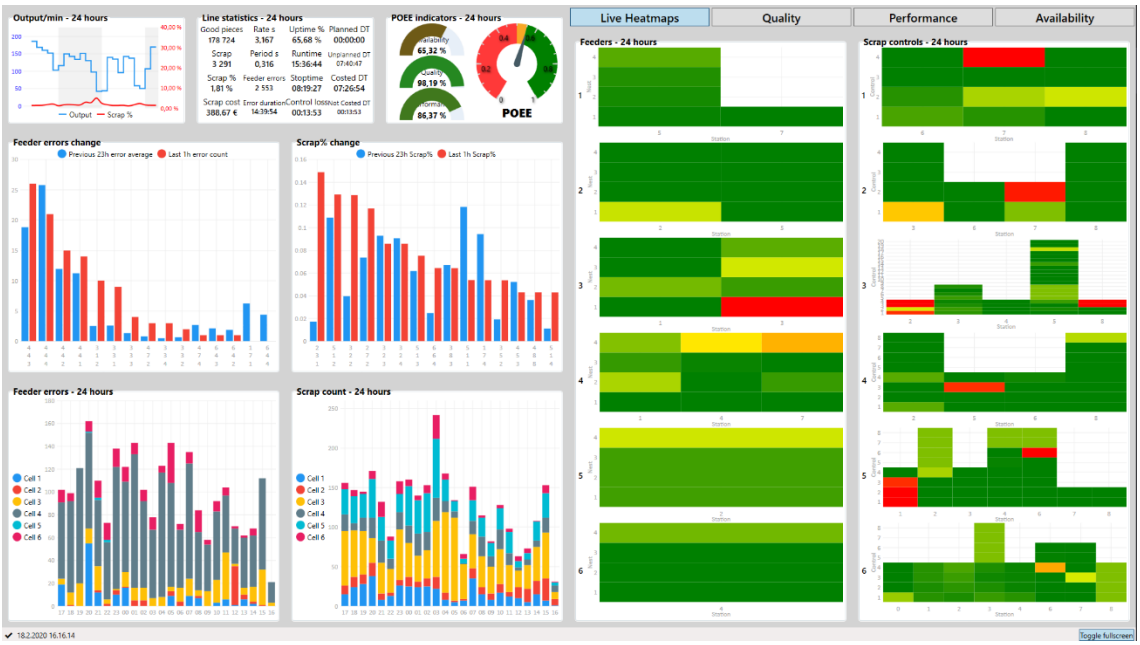
Kuva 7. HeatMapModel-luokka.

6.5 Käyttöliittymä

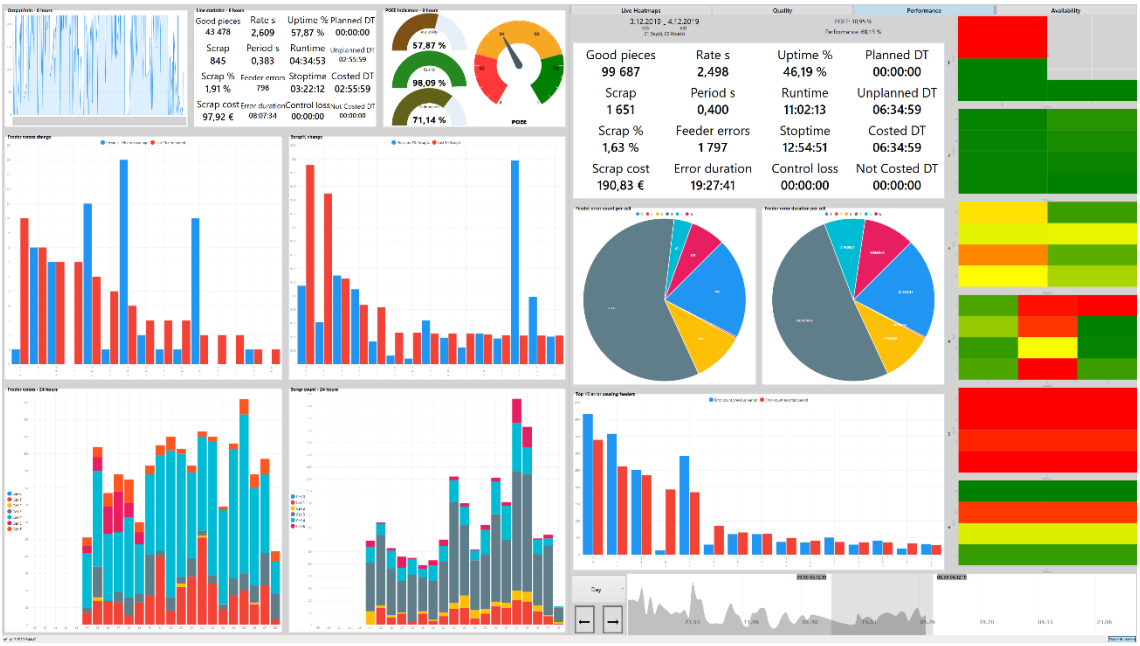
Sovellus on tarkoitettu käytettävän suurelta kosketusnäytöltä (kuva 8), mutta koska se tukee myös hiirikomentoja ja skaalautuu 1080p-resoluutioiselle näytölle, voidaan sitä hyödyntää myös työpöytäkäytössä. Käyttöliittymä voidaan asettaa pystyasentoon asetustiedostoa muokkaamalla.

Kuva 8. Sovellus kosketusnäytöllä.

Sovelluksen käyttöliittymä on jaettu oikeaan ja vasempaan osaan. Vasen puoli näyttää aina kokoonpanolinjan nykytilanteen tiivistetysti. Oikealle puolelle voidaan valita näytettäväksi joko nykyhetken kaikkien laatukontrollien ja syöttölaitteiden nykyhetken lämpökartat (kuva 9) tai ottaa tarkasteluun jokin kokoonpanolinjan toiminnan osakokonaisuus (kuva 10). Kun sovelluksen käyttöliittymä on asetettu pystykuvaan, käyttöliittymä jakautuu ylä- ja alaosiin. Kuvassa 10 käyttäjä on valinnut oikean puolen kielekkeestä tarkasteltavaksi kokoonpanolinjan syöttölaitehäiriöt yhdeltä vuorokaudelta.



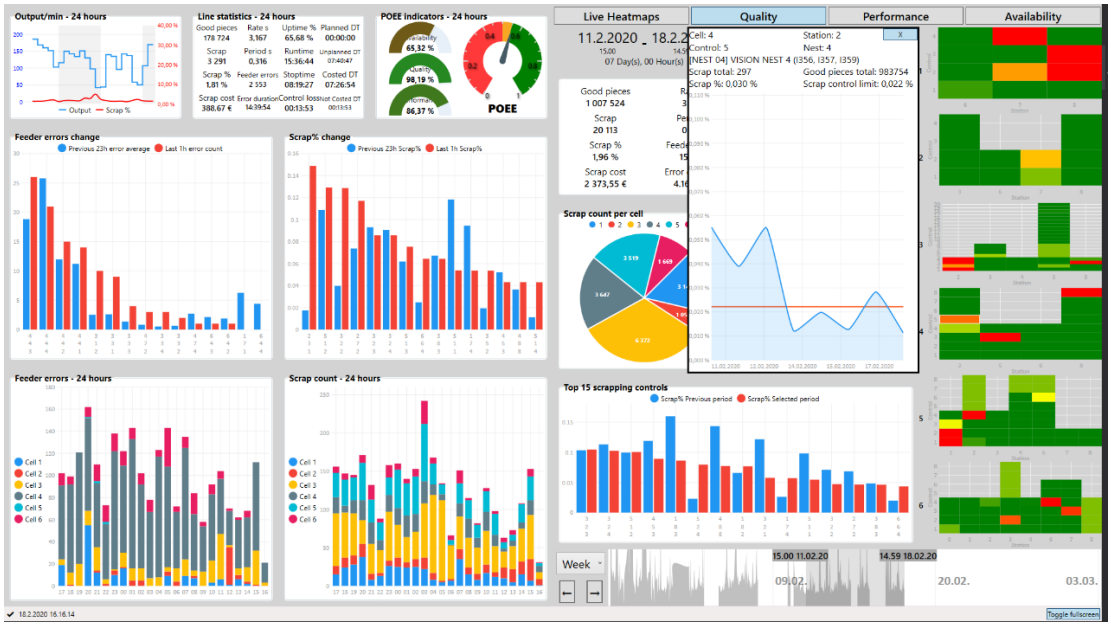
Kuva 9. Sovelluksen aloitusnäkymä skaalattuna tietokoneen näytölle.



Kuva 10. Syöttölaitehäiriöt tarkastelussa.

Tarkastelunäkymien alalaidassa sijaitsee aikavalitsin, joka tukee vieritys-, napauttamis-, nipistämis-, ja venytyskosketuseleitä. Aikavalitsimen taustakuvana toimii sillä hetkellä tarkasteltavan kokonaisuuden virhe- tai häiriösummien kuvaaja. Kaikissa tarkastelunäkymissä on aikavalitsimen lisäksi myös kokoonpanolinjan yleiset tiedot valitulta aikaväliltä.

Kaikkia sovelluksen lämpökarttojen värilaatikoita voi napauttaa avatakseen tiedot, joista väri on muodostettu. Kuvassa 11 on napautettu auki solun 2 aseman 3 laatukontrollin 1 tiedot.



Kuva 11. Laatukontrollien lämpökartan värisolua klikattu.

Ympyrädiagrammien sektoreita napauttamalla käyttöliittymään avautuu kutakin aihealuetta ja sektoria vastaava taulukko, joka näyttää sektorin yksityiskohtaiset tiedot.

Jos havaitaan pitkä pysähdys kokoonpanolinjan toiminnassa, ohjelma pyytää kuittaamaan pysähdyksen syyn, jonka jälkeen ohjelman käyttöä voidaan jatkaa normaalisti. Sovellus hakee valittavat syyt tietokannasta. Konfiguraatiotiedostossa voidaan asettaa aika, joka käyttäjällä on pysähdyksen kuittaamiseen ennen kuin määrittelemätön syy hyväksytään. Pysähdyssyyn käsittely voidaan myös ottaa pois käytöstä.

7 Jatkokehitysehdotukset

Vaikka kokoonpanolinjan hälytystietojen tiedonkeruu ei valmistunut tämän projektin aikana, tietokantaan on määritelty valmiiksi hälytystietojen tallennukseen ja käsittelyyn tarkoitetut taulut ja menetelmät. Sovellukseen hälytystietojen tarkastelunäkymä olisi lisättävissä helposti, koska tarkastelunäkymät jakavat yliluokan, joka sisältää päivitys- ja käyttäjälogiikan. Sovelluksen vasemman puolen yleisnäkymään olisi sovitettavissa yksi kaaviorivi lisää.

Käyttöliittymän elementtien ja kirjasinkoon skaalaus ei toimi täydellisesti pienempiresoluutioisilla näytöillä. Tämän takia osa teksteistä jää graafien taakse. Tähän ei kiinnitetty huomiota sovellusta kehittäessä, sillä sovellusta oli alun perin tarkoitus ajaa vain suurelta kosketusnäytöltä kokonäyttötilassa. Skaalaus täytyy korjata tukemaan eri kokoisia näyttöjä, jos sovellusta tullaan käyttämään perinteisillä tietokoneen näytöillä.

Toimeksiantajalla ilmaisi kiinnostuksen vuororaporttijärjestelmän kehittämiseen sovelluksen sisälle. Tällaisen järjestelmän kehittämisen aloittaminen vaatisi tutkimuksen mahdollisista kerättävistä tiedoista ja niiden muodosta, jotta normalisoitu, hyödyllinen tietorakenne voitaisiin kehittää. Henkilökohtaisesti olen sitä mieltä, että raportointisovelluksen tulisi olla itsenäinen kokonaisuus.

Lämpökartat suunniteltiin alun perin historiatietojen visualisointiin, mutta myöhemmin päätettiin käyttää niitä myös aloitussivulla tämän hetken tilanteen seurantaa varten. Vaikka lämpökartat toimivat hyvin, kun tarkastellaan aikavälien linjan toimintaa, eivät ne täysin sovellu näyttämään reagointitarvetta. Live Heatmaps -välilehden lämpökarttojen arvot olisi mahdollisesti korvattava perinteisten kontrolliarvojen ylityksiä kuvaavilla arvoilla. Tietokantaan olisi tätä varten lisättävä taulurakenne tukemaan kontrolliarvojen varastointia.

Koska pitkien linjaseisokkien syitä alettiin keräämään vasta tämän valvomosovelluksen testausvaiheen myötä, jätettiin Availability-välilehti todella pelkistettyyn tilaan. Ajankäytön mahdollisia esitysmuotoja ovat esimerkiksi vesiputouskaavio,

joissa täydestä aikapotentiaalista vähennetään ryhmitellyt linjan pysähdykset ja joissa jäljelle jää linjan käyntiaika, virheiden tiheyden näkymät ja perinteiset piirakkadiagrammit.

Jos kokoonpanolinjan lyhyiden pysähdyksien määrän vähentäminen nousee kehityskohteeksi, täytyy linjalta alkaa kerätä hälytystietoja. Hälytystietojen avulla voidaan määrittää yleisimmät pysähdyksien syyt. Tällä kokoonpanolinjalla hälytyssyiden kerääminen tietokantaan voi vaatia lisäinvestointeja, joten tämä tuskin on ajankohtaista ennen kuin pitkien suunnittelemattomien linjaseisokkien, hukkaosien ja syöttölaitehäiriöiden määrä on saatu pysymään tavoitetasoissaan.

Koska joidenkin kokoonpanolinjojen rakenne poikkeaa tämän projekti kohdelinjasta huomattavasti, tullaan sovelluksesta tekemään muita versioita uusien linjojen käyttöönoton yhteydessä. Näitä sovellusvarianttiprojekteja ei ruveta suunnittelemaan, ennen kuin tämän projektin sovellus on täysin käyttöönotettu, jotta eri varianttien kehitys ja ylläpito ei vie suunnattomasti resursseja.

8 Pohdinta

Aikaa projektin tekemiseen kului yli 20 työviikkoa suunnitellun kymmenen viikon sijaan. Vaatimusmäärittelyt olisivat täytyneet olla yksityiskohtaisempia. Suunnitteluvaiheessa oletin liikaa, enkä osannut kysyä oikeita kysymyksiä toimeksiantajalta. Näiden oletusten, huonojen vaatimusmäärittelyjen ja ympäristön haasteiden vuoksi yllättäviä ongelmia tuli vastaan huomattavan paljon. Työtä jouduttiin usein tekemään toiseen kertaan, sillä vaatimukset saattoivat muuttua päivittäin. Eniten ylimääräisiä työtunteja aiheuttanut yksittäinen ongelmanlähde oli LiveCharts-visualisointikirjasto. Kun tutkin eri kirjastoja, keskityin suurimmaksi osaksi kirjastontarjoajien esimerkkitapauksiin. Jos olisin tutkinut käyttäjien ongelmatapauksia ja kuinka eri kirjastojen tukipalvelut eroavat toisistaan, olisin huomannut LiveCharts-kirjaston olevan liian keskeneräinen tähän projektiin käytettäväksi. Muutenkin mielestäni projekti oli liian laaja yhden henkilön opinnäytetyöksi, mutta

tämänkin ymmärsin vasta kun projektia toteutettiin. Näiden edellä mainittujen seikkojen vuoksi koen projektin suunnittelun epäonnistuneen.

Vajaasta suunnittelusta huolimatta projektin kaikkiin tavoitteisiin päästiin. Lopputuloksena saatiin toimiva järjestelmä, jolla kokoonpanolinjan toimintaa voidaan seurata tuotantotiloissa. Tietokanta sekä sovellus ovat monistettavissa muille kokoonpanolinjoille. Tämän projektin tietokantamäärittelyn avulla on voitu antaa laitetoimittajille lähtökohdat muiden tuotantolinjojen tiedonkeruun suunnittelun pohjaksi.

Toimeksiantaja on lopputulokseen tyytyväinen. Järjestelmä on testikäytössä kohdelinjalla ja sen on huomattu vastaavan hyvin toimeksiantajan tarpeita. Sovellusta jatkokehitetään ja tietokantaa ollaan monistamassa muille kokoonpanolinjoille. Yrityksessä nyt olevan yhtenäisen tietokantamäärittelyn avulla on voitu esittää linjatoimittajille vaateita uusien linjojen tiedonkeruusta. Mielestäni tämä projekti näytti toimeksiantajalle, että on mahdollista toteuttaa ohjelmistoprojekteja pienellä työryhmällä ilman ulkopuolisia resursseja mahdollistaen tarpeeseen äärimmilleen räätälöidyn lopputuloksen.

Tämä projekti kehitti henkilökohtaista osaamistani usealla eri osa-alueella. C#, MVVM ja WPF tulivat luonnollisesti tutummiksi ja tietokantaosaamiseni kehittyi huomattavasti. Siitä huolimatta, että tämä opinnäytetyö tuntui aluksi teknisen osaamisen näytöltä, tekninen osaaminen ei ollut mielestäni tärkein oppimisen kohde. Ymmärrän paremmin suunnittelun ja asioiden tarkasti ennalta selvittämisen hyödyn. Olen huomannut tämän projektin jälkeen tilanteita muissa projekteissa, joissa ennen olisin olettanut asioiden laidan, mutta joissa nyt ymmärrän kysyä lisätietoja. Projektinhallinnassa minulla on vielä paljon kehitettävää ja pyrin tulevaisuudessa oppimaan rutiininomaisemmat käytännöt koko projektin elinkaarelle. Projekti, jossa toimeksiantajan puolella ei ollut syvää tietoteknistä osaamista, oli tärkeä kokemus, koska kyky ymmärtää ja tulla ymmärretyksi keskustelussa eri alan asiantuntijan kanssa on huomattavan tärkeää projektin sujuvuuden kannalta.

Minut yllätti, miten tuotantoympäristössä verrattain pienilläkin projekteilla voidaan saavuttaa valtava hyöty. Tietotekniikan koulutusohjelmaan olisi ehkä hyvä kuulua myös massatuotantoympäristöön tutustumista.

Lähteet

1. Microsoft. Overview of Entity Framework 6 – EF6. Microsoft. 2016. Päivitetty 5.10.2019. [Viitattu 23.1.2020]. Saatavissa: https://docs.microsoft.com/en-us/ef/ef6/.
2. Villanen, H. Tuotantokoneiden kokonaistehokkuus, OEE (Overall Equipment Efficiency). Prosessitaito Hannu Villanen. 2013. Saatavissa: http://www.prosessitaito.fi/Tuotantokoneiden_kokonaistehokkuus_OEE.pdf.
3. Microsoft. Introduction to WPF. Microsoft. 2016. Päivitetty 21.1.2020. [Viitattu 15.2.2020]. Saatavissa: https://docs.microsoft.com/en-us/dotnet/framework/wpf/introduction-to-wpf.
4. Microsoft. Use a SQL Server database in a UWP app. Microsoft. 2019. Päivitetty 5.10.2019. [Viitattu 3.12.2019]. Saatavissa: https://docs.microsoft.com/en-us/windows/uwp/data-access/sql-server-databases.
5. Smith, J. Patterns – WPF Apps With The Model-View-ViewModel Design Pattern. MSDN Magazine. Volume 24 Number 2. 2009. [Viitattu 12.11.2019]. Saatavissa: https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern.
6. LiveCharts. Live Charts. lvcharts.net. 2020. [Viitattu 4.3.2020]. Saatavissa: https://lvcharts.net/.
7. Lim, R. A Simple MVVM Example. Rachel Lim's Blog. 2011. [Viitattu 4.3.2020]. Saatavissa: https://rachel53461.wordpress.com/2011/05/08/simplemvvmexample/.