TAMPEREEN TEKNILLINEN YLIOPISTO

TURO KORHONEN

TYÖVAIHEISIIN KULUVAN AJAN SEURANNAN KEHITTÄMINEN ELEKTRONIIKKA- ALAN YRITYKSESSÄ

Diplomityö

Tarkastajat: tutkija Anne-Maritta Tervakari ja dosentti Ossi Nykänen
Tarkastajat ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 5. toukokuuta 2014

TIIVISTELMÄ

TURO KORHONEN: Työvaiheisiin kuluvan ajan seurannan kehittäminen elektroniikka-alan yrityksessä

Tampereen teknillinen yliopisto

Diplomityö, 65 sivua, 1 liitesivua

Toukokuu 2015

Tietotekniikan koulutusohjelma

Pääaine: Hypermedia

Tarkastajat: tutkija Anne-Maritta Tervakari ja dosentti Ossi Nykänen

Avainsanat: tuotannon seuranta, tuottavuus, tuotanto

Tämä työ on tehty Kyrel Oy:lle. Kyrel Oy on alihankintayritys, joka tarjoaa elektroniikka-, mekaniikka- ja sähkömekaniikan kokoonpanopalveluja pitkällä kokemuksella. Kyrel Oy:n ja monen muun yrityksen haasteena on ollut, ettei tarkkaa tietoa siitä, kauanko tietty työvaihe kestää milläkin tuotteella ole pystytty keräämään helposti. Työvaiheissa tehtävien töiden määrä ja niiden pituus saattavat vaihdella paljon eri tuotteiden välillä aiheuttaen hankaluuksia työvaiheiden keston mittaamiselle.

Työvaiheiden kestoa on arvioitu mittaamalla sekuntikellolla ja laskemalla erilaisten matemaattisten kaavojen mukaisesti. Näiden avulla ei ole kuitenkaan pystytty keräämään kaikkia tietoja todellisista työn kestoista, koska työvaiheita suoritettaessa ilmenee myös muita pakollisia tehtäviä, joiden vaikutusta työn suorittamiseen kuluvaan aikaan ei pystytä tarkasti arvioimaan. Tällaisia ovat esimerkiksi työohjeiden etsiminen ja tutkiminen, komponenttien haku ja erilaiset dokumentoinnit. Työvaiheiden tehtävät ja työskentelytavat ovat saattaneet myös muuttua, eikä muutoksia ole mitattu. Yritys voi hyödyntää tietoa yhden työvaiheen kestosta kullakin tuotteella esimerkiksi työn hinnoittelussa ja tuotannon suunnittelussa.

Tässä työssä tutkittiin aluksi, löytyisikö valmiita ratkaisuja ongelman selvittämiseksi. Hyvin pian kävi kuitenkin selville, ettei halutunlaista järjestelmää oltu esitelty avoimesti. Tämän jälkeen alettiin kerätä ideoita tekniikoista, joilla tuotannon seurannan voisi toteuttaa. Kerättyjen tietojen pohjalta tehtiin suunnitelma toteutettavasta järjestelmästä ja aloitettiin suunnitelman mukainen toteutus. Valittu järjestelmä pohjautui tablet-tietokoneen kautta tapahtuvaan tietojen syöttöön. Työvaiheen tiedot luettiin työkortin viivakoodista tabletin kameran avulla. Työkortti oli A4-kokoinen paperi, joka sisälsi kyseisen työn erä-, tuote- ja työvaihetiedot. Työntekijän tehtävänä oli viivakoodin lukemisen lisäksi merkata järjestelmän avulla työn aloitus- ja lopetusaika. Työssä päästiin toiseen testausvaiheeseen, jonka avulla järjestelmä todettiin toimivaksi.

ABSTRACT

TURO KORHONEN: Development of monitoring the time taken in stages in an electronics company

Tampere University of Technology

Master of Science Thesis, 65 pages, 1 Appendix pages

May 2015

Master's Degree Programme in Information Technology

Major: Hypermedia

Examiner: researcher Anne-Maritta Tervakari and adjunct professor Ossi Nykänen

Keywords: supervising of production, productivity, production

This thesis in made for Kyrel Oy. Kyrel Oy is a subcontractor offering electronics, mechanics and electromechanic assembly services with long experience.The challenge of Kyrel Oy and many other companies has been that they haven't been able to collect information about how long a certain stage takes time with each product. The number and duration of stages may vary a lot between different products causing difficulties for measuring the duration of stages.

The duration of each stage has been estimated by stopwatch and by measuring with different sorts of formulas. Even with these measurements, it hasn't been possible to collect the exact information about the duration of stages, because during stages, other mandatory tasks come up and the effect of these tasks for the duration cannot be measured accurately. Such tasks are for example searching and researching instructions, fetching of components and different documentations.

The assignments and methods of stages could also have changed and the changes haven't been measured. The company can use the information of duration of one stage with each product for example in pricing and in production planning.

First in this thesis the possibility for solutions already made for the problem was researched. It was soon clear that the system of the specified type hasn't been openly presented. After this the collecting of ideas for techniques for the supervising of production began. Per the collected ideas a layout was made for the system and the implementation according to the layout began.

The system chosen is based on the information input via a tablet computer.

The stage information was read from the barcode of the job card by the camera of the tablet computer. The job card was an A4-sheet containing the information about assignment's batch, product and stage. The assignment for the user, besides the reading of the barcode, was to mark the beginning and ending times. In this thesis the second testing phase was achieved and the system was stated functional.

ALKUSANAT

Tämä työ oli hyvin monipuolinen ja opettavainen. Työtä tehdessä oppi monia uusia asioita ja pääsi toteuttamaan käytännössä monia luennoilla opittuja tietoja. Haluaisin kiittää Kyrel Oy:tä mahdollisuudesta diplomityön tekemiseen. Erityiset kiitokset kehityspäällikkö Ville Salomäelle ja kaikille muille, jotka auttoivat tämän projektin läpiviennissä. Haluaisin myös kiittää avovaimoani Hannaa ajasta, jonka hän järjesti minulle tämän työn kirjoittamista varten. Terveiset myös lapsilleni Ilonalle, Aaronille ja Topiakselle. Toivon, että sitten joskus tätä lukiessanne ymmärrätte hetkiä, jolloin isi teki ”kouluhommia”, eikä häntä saanut häiritä. Kiitos myös siskolleni Katjalle, joka toimi oivallisena kirjoitusvirheiden metsästäjänä.

Isot kiitokset myös diplomityön tarkastajille Anne-Maritta Tervakarille ja Ossi Nykäselle, jotka mahdollistivat työn valmiiksisaattamisen lyhyellä varoitusajalla.

Kangasalla 17.5.2015

Turo Korhonen

SISÄLLYS

1 JOHDANTO....1
1.1 Yrityksen esittely....2
1.2 Työn rakenne....2
1.3 Ongelman esittely....2
1.3.1 Pääongelma....3
1.3.2 Mittauksen ongelmia....3
1.4 Lähtöasetelma....4
2 ESISELVITYS....6
2.1 Ensimmäinen vaihe....6
2.1.1 Materiaalikulun seuranta....6
2.1.2 Työtehtävien kirjaus....7
2.1.3 Lopputulos....7
2.2 Tuotannon seurantaan sopivia tekniikoita....8
2.2.1 Viivakoodi....8
2.2.2 Kaksiuloitteiset viivakoodit....12
2.2.3 RFID (Radio Frequency IDentification)....18
2.2.4 Konenäkö....20
2.3 Välineitä ongelman ratkaisemiseksi....21
2.3.1 RFID-tunniste tai viivakoodi ja käyttäjän syöte....22
2.3.2 Integroidulla RFID- tai viivakoodinlukijalla varustettu kosketusnäyttölaite....23
2.4 Ensimmäisen osion yhteenveto....25
3 KÄYTETTÄVYYS....27
3.1 Käytettävyyden määritelmä....27
3.1.1 Käytettävyys ISO-9241-11 mukaisesti....28
3.1.2 Käytettävyys Nielsenin mukaan....29
3.2 Käytettävyyden arviointi....30
3.2.1 Heuristinen analyysi....30
3.2.2 Käytettävyystestaus....33
3.3 Käyttökokemus tai käyttäjäkokemus....34
4 OMAN RATKAISUN ESITTELY....36
4.1 Back-end toteutus....39
4.1.1 Open Database Connectivity (ODBC)....39
4.1.2 SQL....40
4.1.3 Internet Information Services (IIS)....41
4.1.4 Fast Common Gateway Interface (FastCGI)....41
4.1.5 PHP....42
4.2 Front-end toteutus....42

5 ENSIMMÄINEN VERSIO....44
5.1 Ensimmäinen prototyyppi....44
5.2 Toinen prototyyppi....44
5.3 Näkymät....45
5.3.1 Työntekijänumeron syöttäminen ja ohjesivu....45
5.3.2 Töiden tietojen syöttäminen ja avoimet työt....47
5.4 Ensimmäisen version testaus....50
5.4.1 Testauksen havainnot....51
5.4.2 Testaus muilla laitteilla....52
6 TOINEN VERSIO....53
6.1 Työkortin jatkokehitys....53
6.2 Sovelluksen toinen versio....55
6.3 Näkymät....56
6.3.1 Työntekijänumeron syöttäminen ja ohjesivu....56
6.3.2 Töiden tietojen syöttäminen ja avoimet työt....57
6.4 Toisen version testaus....60
6.4.1 Sovelluksen käytettävyyden arviointi....61
6.4.2 Testauksen aikana syntyneet huomiot....63
7 LOPPUTULOS....65
LÄHTEET....66

LIITE A: ENSIMMÄISEN TESTIVERSION KYSELYLOMAKE

TERMIT JA NIIDEN MÄÄRITELMÄT

Tuotannon mittareita

Apuaika	Aika joka varataan henkilökohtaisia tarpeita varten, esim kahvitauot
Häiriöaika	Aika, joka kuluu kaikenlaisiin odottamattomiin keskeytyksiin
Jaksonaika	Kertoo, kauanko ihmiseltä menee aikaa tietyn työvaiheen suorittamiseen
Koneaika	Kertoo kauanko koneella menee aikaa tietyn työn tekemiseen
Päivävakio	Arvio siitä kauanko päivän työtunneista menee sellaiseen työhön mitä ei voida kohdistaa suoraan tuotteen valmistamiseen
Solun läpimenoaika	Kertoo kauanko tuotteen valmistus kestää tietyssä solussa
Tahtiaika	Kertoo ajan, joka yhden tuotteen valmistukseen saa kulua, jotta kysyntä pystyttäisiin täyttämään.
Toiminnanohjaus-järjestelmä	Yrityksen tietojärjestelmä, joka yhdistää monia yrityksen toimintoja samaan järjestelmään

Tekniikoita

Aztec Code	Kaksiuloitteinen viivakoodimuoto, jonka tunnistaa pyramidia muistuttavasta kohdistusmerkistä.
CCD-kenno	Charge-Coupled Device. Valoherkkä kenno, joka koostuu fotodiodeista.
CMOS	Complementary Metal Oxide Semiconductor. Kanavatransistoreihin perustuva tekniikka.
Code 128	Viivakoodimuoto joka sisältää 128 merkkiä, viivakoodissa myös tarkistussumma virheiden löytämiseksi
Code 39	Viivakoodimuoto joka sisältää 43 eri merkkiä
Data Matrix	Kaksiuloitteinen viivakoodimuoto, joka yleisessä käytössä teollisuudessa.
EAN	Viivakoodimuoto jota käytetään vähittäiskaupassa tuotteiden tunnistamiseen.
ECC 200	Data Matrixin versio
Fotodiodi	Valolle herkkä diodi
HF	High Frequency. 3-30 Mhz
LF	Low Frequency. 30-300 kHz

Lähikenttä UHF	Tapa jolla parannateen UHF-taajuuksilla toimivien RFID-järjestelmien toimintaa metallien ja nesteiden lähellä. Perustuu magneettikenttään.
Mini-ITX	Erittäin pieni emolevyjen kokoluokka.
NFC	Near Field Communication. RFID tekniikkaa hyväksikäyttävä tapa, jolla voidaan toteuttaa esimerkiksi matkapuhelimella tapahtuva mobiilimaksaminen
OTG	On-The-Go. Tekniikka, jolla USB-laite voi toimia isäntänä USB-väylällä
QR-koodi	Quick Response Code. Kaksiuloitteinen viivakoodimuoto, joka on nykyään suosittu matkapuhelinsovelluksissa.
Raspberry Pi	Yhden piirilevyn pienikokoinen tietokone.
RFID	Radio Frequency Identification. Yleisnimitys radiotaajuuksilla toimiville tekniikoille, joita käytetään esimerkiksi tunnistamiseen.
Tablet	Taulutietokone. Kosketusnäytöllä ohjattava tietokone.
UHF	Ultra High Frequency. Taajuusalue 0,3–3 GHz
USB	Universal Serial Bus. Sarjaväyläarkkitehtuuri oheislaitteiden liittämiseksi.

Tietotekniikka

Android	Puhelimille ja tableteille kehitetty käyttöjärjestelmä
AppGyver Steroids	Sovelluskehys hybridimobiilisovellusten tekoon.
Back-end	Palvelimella ajettava web-sovellus.
Bluetooth	Avoin standardi laitteiden langattomaan kommunikointiin.
Cordova	Sovelluskehys hybridimobiilisovellusten tekoon.
CSS	Cascading Style Sheets. Tyyliohje, käytetään www-sivujen muotoiluun.
Eclipse	Ohjelmistoympäristö
FASTCGI	Ohjelma minkä avulla voidaan ajaa toisia ohjelmia web-sivun kautta
Frond-end	Käyttäjän päässä ajettava sovellus, yleensä tarkoitetaan JavaScript-koodia, joka ajetaan selaimessa.
GET-pyyntö	Tapa ladata web-sivu niin, että pyynnön ohessa lähetetään parametreja osoitteen mukana.

HTML	Hypertext Markup Language. Hypertekstin merkintäkieli. Käytetään www-sivujen teossa.
HTML 5	Uusin versio verkkosivujen tekemiseen käytetystä HTML-merkintäkielestä
IIS	Internet Information Services. Microsoftin kehittämä palvelinohjelmistokokonaisuus.
Java	Ohjelmointialusta johon kuuluu muun muassa ohjelmointikieli, virtuaalikone ja luokkakirjasto.
JavaScript	Komentosarjakieli. Yleinen web-sivuilla.
JSON	JavaScript Object Notation. Avoimen standardin tiedostomuoto tiedonvälitykseen.
Local storage	Tekniikka, jolla tallennetaan tietoja internet-selaimeen
Natiivi-sovellus	Jokaiselle eri laitealustalle erikseen ohjelmoitu sovellus
ODBC	Open Database Connectivity. Avoin rajapinta tietokantojen ja ohjelmien väliin.
PhoneGap	Sovelluskehys hybridimobiilisovellusten tekoon.
PHP	Komentosarjakieli, jota käytetään web-palvelinympäristössä.
POST-pyyntö	Tapa jolla lähetetään web-palvelimelle tietoja www-sivun lomakkeelta niin, että tieto välitetään HTML-pyynnön otsikkotietojen mukana.
SQL	Structured Query Language. Standardoitu kyselykieli relaatiotietokantojen käsittelyyn.
WebView	Android-käyttöjärjestelmän luokka, jolla voidaan ladata web-sivu osaksi Android sovellusta.
Zxing	Avoin Java-kirjasto viivakoodien käyttöön.
Heuristiikat	Kokoelma periaatteita tai sääntöjä.
ISO 9241-210	Standardi ihmiskeskeisen suunnittelun periaatteista
ISO-9241-11	Käytettävyyden standardi
Käytettävyystestaus	Käytettävyyden testausmuoto, jossa seurataan todellisen käyttäjän toimia

1 JOHDANTO

Tässä diplomityössä tutustutaan keinoihin, joilla elektroniikka-alan yrityksen työvaiheisiin kuluvan ajan seurantaa voisi parantaa. Aluksi tutkitaan, millä tavalla seuranta on toteutettu tällä hetkellä ja tämän jälkeen mietitään eri keinoja, joilla sitä voitaisiin parantaa. Mietinnän jälkeen tehdään suunnitelma seurannan toteuttamiseksi toteutuskelpoisimman ratkaisun perusteella. Seurannan pitäisi onnistua nopeasti ja luotettavasti, jottei se hidasta varsinaisen työn tekoa. Järjestelmän pitäisi olla tarpeeksi edullinen, jotta se voitaisiin ottaa käyttöön joka työpisteellä. Lisäksi sen pitäisi olla käytettävyydeltään niin hyvä, että sen käyttö onnistuu kaikilta työntekijöiltä. Suunnittelun pohjalta rakennetaan järjestelmä ja testataan sen toimintaa sekä tehdään tarpeelliset muutokset testijakson aikana syntyneiden huomioiden perusteella.

Tuottavuuden mittaamiseen on kehitetty erilaisia laskentakaavoja ja tapoja. Tuottavuutta selvitettäessä tarkastellaan tuotteiden valmistukseen kuluvia resursseja ja niiden suhdetta saatuun tulokseen. Tuottavuutta mitataankin tuotoksen suhteella panokseen. Tuottavuutta voidaan tarkastella monella tasolla aina koko organisaation tai maan tuottavuudesta yhden työvaiheen tai työntekijän tuottavuuden tarkasteluun. Kuitenkin yleisesti on hyvin vähän saataville tietoa siitä, minkälaisilla mittaus- tai seurantavälineillä voitaisiin kerätä tietoa yhden työvaiheen tai tuotteen valmistukseen kuluvasta ajasta. Tämän työn ensisijainen tarkoitus onkin kehitellä väline, jolla voitaisiin seurata työvaiheessa kuluvaa aikaa niin, ettei se hankaloita liikaa itse työtä eikä aiheuta itse tuottavuuden vähenemistä. Tämän tiedon avulla pystyttäisiin keräämään tarkkaa tietoa tuotteiden ja työvaiheiden kestosta, jolloin ne voidaan hinnoittella oikein. Tämä vaikuttaa suoraan yrityksen kannattavuuteen.

1.1 Yrityksen esittely

Diplomityö tehtiin Kyrel Oy:lle, joka on elektroniikan, mekaniikan ja sähkömekaniikan kokoonpanopalvelua tarjoava yritys. Yritys sijaitsee Kyröskoskella noin 40 kilometrin päässä Tampereelta. Yritys on alunperin perustettu 1978. Vuosituhannen vaihteessa Flextronics International Ltd osti Kyrelin. Flextronics toimi Kyröskoskella muutaman vuoden ja vuonna 2003 ilmoitti luopuvansa Kyröskosken tehtaasta, jolloin Kyrel Oy perustettiin uudestaan. Vanhat omistajat ostivat Flextronicsilta laitteet ja materiaalit, jolloin tehtaan toiminta jatkui. Yritys työllistää tällä hetkellä noin 150 henkilöä. Asiakkaina on yli 50 yritystä ja tuotteita tuhansia erilaisia. Valmistus tapahtuu soluissa, jotka muodostuvat asiakas- ja työvaiheperusteisesti.

1.2 Työn rakenne

Diplomityö jakautuu kolmeen osaan. Ensimmäisessä osassa esitellään ongelma ja kerrotaan sen haasteista. Toisessa osassa, eli luvussa 2 tutkitaan olemassa olevia valmiita ratkaisuja ongelman ratkaisemiseksi ja mietitään tekniikoita, joita voisi hyödyntää. Samalla tutkitaan yrityksessä käytössä olevia keinoja tuotannon seurantaan sekä käydään läpi teoriaa näiden pohjalta. Kolmannessa osassa suunnitellaan ja esitellään ratkaisu, joka tuntui toteutuskelpoisimmalta niin tarpeiden, kustannusten kuin toteutettavuudenkin perusteella. Tämän jälkeen kerrotaan järjestelmän toteutuksesta, testausvaiheesta ja testauksesta saatujen tietojen avulla tehdyistä muutoksista. Tässä työssä keskitytään tapaan, jonka avulla tietoa voitaisiin kerätä, ei niinkään siihen, miten kerättyä tietoa voitaisiin hyödyntää.

1.3 Ongelman esittely

Tässä luvussa kerrotaan työhön liittyvistä ongelmista. Ensimmäiseksi kerrotaan pääongelmasta ja tämän jälkeen mittauksien yleisistä ongelmista.

1.3.1 Pääongelma

Haasteena on, ettei tuotannon työvaiheista ole pystytty keräämään helposti tarkkaa tietoa siitä, kauanko tietty työvaihe kestää milläkin tuotteella. Työvaiheiden kestoa on pyritty arvioimaan mittaamalla sekuntikellolla ja laskemalla erilaisten matemaattisten kaavojen mukaisesti. Näiden avulla ei ole kuitenkaan pystytty keräämään tarkkaa tietoa todellisista työn kestoista, koska työvaiheita suoritettaessa syntyy myös muita pakollisia tehtäviä, joiden vaikutusta työn suorittamiseen kuluvaan aikaan ei pystytä kokonaan huomioimaan. Tällaisia ovat esimerkiksi työohjeiden etsiminen ja tutkiminen, komponenttien haku ja erilaiset dokumentoinnit. Työvaiheiden työt ja työskentelytavat ovat saattaneet myös muuttua eikä muutoksia ole mitattu. Tällaisia ylimääräisiä työn suorittamiseen vaikuttavia asioita ei pystytä mittaamaan lyhyillä seurantajaksoilla, eikä niiden todellista vaikutusta pystytä arvioimaan tarpeeksi tarkasti. Tätä ongelmaa varten pitäisi löytää väline, jolla työvaiheiden pitemmän ajan seuranta onnistuisi.

1.3.2 Mittauksen ongelmia

Yleisesti mittauksen suurimpia ongelmia voidaan mainita kaksi. Ensimmäiseksi täytyy tietää mitä mitataan. Oikeanlaisia mittauksia on hankala suorittaa, jos ei tiedetä mitä oikeastaan pitäisi mitata. Tämän takia täytyy aluksi miettiä, mitä tietoja halutaan kerätä, ja vasta tämän jälkeen aletaan miettiä, miten haluttu tieto voitaisiin mitata. Toinen hyvin tärkeä seikka mitattaessa on se, ettei mittaustapahtuma saisi häiritä mitattavaa tapahtumaa. Jos mittaustapahtuma vaikuttaa mitattavaan, ei tällöin saada mitattaessa oikeita tuloksia.

Työvaiheiden kestot vaihtelivat paljon, muutamasta sekunnista tunteihin, mediaanin ollessa noin 3-5 minuuttia. Eräkoot saattoivat vaihdella tyypillisesti muutamasta kappaleesta tuhansiin. Hyvin erilaiset työvaiheet ja eräkoot aiheuttivat haasteita työaikojen mittaamiselle. Työn lähtökohtana oli kuitenkin kehitellä väline, joka sopisi mahdollisimman monen työvaiheen seurantaan.

1.4 Lähtöasetelma

Yrityksessä seurattiin ladontakoneiden, useiden testauslaitteiden ja monen muun laitteen läpimenevien tuotteiden määrää. Tällä tavoin voidaan seurata jonkin laitteen tai koneen koneaikaa (Machine time MT). Tämä lasketaan kaavan (1) mukaisesti. Koneaika kertoo, kauanko koneella tai laitteelle menee sen itsenäiseen työhön. Koneaika kertoo siis ajan, joka tuotantolaitteelta kuluu oman työvaiheensa suorittamiseen. Tämä aika ei kuitenkaan kerro kuinka nopeasti yksi tuote valmistuu kyseisessä solussa, koska koneen käyttöön liittyy myös työntekijän manuaalinen työ. Tämän manuaalisen työn tekemiseen kuluvaa aikaa sanotaan jaksonajaksi (cycle time CT). Jaksonaika saadaan laskemalla kaavalla (2). Tahtiaika (Takt Time TT) kertoo taas sen, millä nopeudella yhden tuotteen pitäisi valmistua, jotta kysyntä pystyttäisiin täyttämään. Tahtiajan laskenta tapahtuu kaavan (3) avulla. Esimerkiksi: asiakkaan tarve tuotteelle on 1000 kappaletta vuorokaudessa. Vuorokaudessa on 8 tuntia, joista 30 minuuttia menee taukoihin ja töitä tehdään kahdessa vuorossa. Tällöin yhden tuotteen valmistamiseen on käytettävissä 54 sekuntia. [9,35]

$$Koneaika = \frac{koneen\ tekemien\ tuotteiden\ määrä}{koneen\ käyttämä\ aika} \tag{1}$$

$$Jaksonaika = \frac{valmistetut\ tuotteet}{käytetty\ työaika} \tag{2}$$

$$Tahtiaika = \frac{työaika}{tarve} \tag{3}$$

Solun läpimenoajalla (ThroughPut Time TPT) tarkoitetaan sitä aikaa, jonka tuote on solussa [35]. Läpimenoaikaan lasketaan kaikki työvaiheet joiden läpi tuote kulkee. Solun läpimenoaika sisältää siis kaikki kyseiseen soluun liittyvien koneiden koneajat ja kaikki työntekijöiden tekemien töiden ajat. Tässä projektissa kehitettävän järjestelmän olisi pystyttävä mittaamaan tuotteiden läpimenoaikoja eri työvaiheissa.

Yrityksessä oli laskettu työaikoja seuraamalla työvaiheiden tekoa, arvioimalla työhön kuluvaa aikaa aikaisempien kokemuksien pohjalta sekä käyttämällä vakioita

tietynlaiseen työhön. Näiden kerättyjen tietojen, päivävakion ja elpymisaikojen avulla laskettiin työvaiheeseen mahdollisesti kuluva aika. Päivävakiolla arvioidaan sitä aikaa, joka työpäivästä kuluu sellaiseen työhön, jota ei voida kohdistaa suoraan tuotteen valmistamiseen. Tällaisia on esimerkiksi työpaikan kunnossapito, työkorttien raportointi ja koneiden säännöllinen huoltaminen. Henkilökohtainen apuaika on aika, joka varataan henkilökohtaisia tarpeita ja työstä johtuvasta kuormituksesta elpymistä varten. Esimerkiksi sovitut kahvitauot ovat henkilökohtaista apuaikaa. Myös häiriöaika vähentää tuottavan työn osuutta käytettävästä ajasta. Häiriöaikoihin lasketaan erilaiset keskeytykset, aputyöt ja odotukset, jotka ovat odottamattomia, eikä niiden esiintymistiheyttä tiedetä etukäteen. Häiriöaikaan kuuluu myös kaikki turha työ, kuten virheiden korjaukset. [35]

2 ESISELVITYS

Tämän esiselvityksen avulla pystyttiin kartoittamaan, mitä valmiita ratkaisuja oli olemassa ja kuinka ne soveltuisivat tarkoitettuun käyttöön. Alaluvuissa kerrotaan, miten valmiita ratkaisuja etsittiin, mitä löydettiin ja lopuksi etsinnän lopputulos.

2.1 Ensimmäinen vaihe

Aluksi lähdettiin etsimään valmiita ratkaisuja ongelmaan. Valmiita ratkaisuja lähdettiin etsimään yleisesti esillä olevista lähteistä, eikä tässä vaiheessa lähdetty kysymään tarjouksia räätälöidylle järjestelmälle.

Löydetyt ratkaisut perustuivat joko materiaalien kulun seurantaan ja hallintaan tai työtehtävien kirjaukseen. Tällaisia ratkaisuja ovat esimerkiksi Trackway Manufacturing [36]. Tämä ratkaisu on Trackway Oy:n kehittämä järjestelmä tuotannon tehostamiseen. Se pohjautuu tunnisteisiin, jotka voivat olla viivakoodeja, kaksiulotteisia viivakoodeja tai RFID-tunnisteita. Toisena esimerkkinä voisi mainita AitoSolutions Oyj:n kehittämän työajanhallinta-ohjelmiston [2]. Se perustuu järjestelmään, jossa aluksi määritellään kaikki työaikaan ja työvuoroihin liittyvät asetukset. AitoID-tiedonkeruumoduulin avulla voidaan tuotannon tiedot kerätä leimauspäätteen kautta joko viivakoodeista tai RFID-tunnisteista.

2.1.1 Materiaalikulun seuranta

Materiaalikulun seurantajärjestelmät on yleensä suunniteltu yhden isomman kappaleen tai pienempien kappaleiden, laatikoiden, lavojen ja muiden isompien pakkauksien

seurantaan. Niiden perusteella olisi kuitenkin työlästä seurata yhden kappaleen liikettä tuotannossa, koska erä ei kulje välttämättä kokonaisena tuotannon eri soluista toisiin. Lisäksi tällaisia valmiita ratkaisuja ei oltu kehitetty työajan seurantaan.

2.1.2 Työtehtävien kirjaus

Työaikojen seurantajärjestelmät perustuvat usein joko valmiiden taulukoiden täyttämiseen tai viivakoodin avulla tapahtuvaan leimaukseen. Tälläiset järjestelmät ovat kuitenkin tarkoitettu enemmän työntekijöiden tekemien töiden seurantaan kuin työtehtävien kestojen seurantaan.

Näiden ratkaisujen ongelma on yleensä se, ettei niiden avulla saada tarpeeksi tarkkaa tietoa siitä, kuinka kauan yhden kappaleen valmistukseen menee oikeasti työaikaa. Lisäksi raportointi tapahtuu usein tietyllä tietokoneella tai muulla laitteella., jolloin työpisteeltä joudutaan aina siirtymään raportointia varten muualle. Tällöin siirtyminen kuluttaa turhaa työaikaa ja vaatii työntekijältä raportoinnin muistamista. Tällaiset järjestelmät muistuttavat suuresti jo yrityksen käytössä olevaa erien raportointia tuotannonohjausjärjestelmään, joten ne eivät suoraan toisi mitään uutta ongelman ratkaisemiseen.

2.1.3 Lopputulos

Etsinnöissä ei löytynyt avointa ja vapaassa levityksessä olevaa ratkaisua, joka olisi yhdistänyt nämä kaksi aikaisemmin esiteltyä tapaa. Kaupallisten ratkaisujen räätälöinti tai uuden järjestelmän teettäminen joltain sopivalta taholta olisi voinut ratkaista ongelman, mutta niiden kustannukset olisivat saattaneet olla tässä vaiheessa liian isoja. Kustannustehokkaampaa oli siis tässä vaiheessa lähteä kehittämään järjestelmää itse. Projekti jatkui seuraavaksi vaiheella, jossa sopivia tekniikoita järjestelmän toteuttamiseksi alettiin etsiä.

2.2 Tuotannon seurantaan sopivia tekniikoita

Seuraavissa alaluvuissa käydään läpi kaikki löydetyt tavat, joita voitaisiin käyttää tuotannonseurannan apuna. Tärkeimmistä tavoista kerrotaan tarkemmin ja pohditaan niiden sopivuutta tähän projektiin. Koska viivakoodit ovat olleet käytössä monia vuosia ja niitä on kehitetty todella monen muotoisia moneen tarkoitukseen, oli järkevintä lähteä tutkimaan ensiksi viivakoodien mahdollisuuksia. Luvussa kerrotaan yleisimmistä viivakoodityypeistä, viivakoodinlukijoista ja aluksi viivakoodin historiasta. Tämän jälkeen esitellään yleisimmät kaksiuloitteiset viivakoodit, joista voisi löytyä sopiva tekniikka tuotannon seurantaan. Seuraavaksi kerrotaan RFID-tunnisteista ja konenäöstä, koska näiden hyödyntäminen projektissa saattaisi olla myös mahdollista. Lopuksi kerrotaan tekniikoista joihin päädyttiin, ja syy miksi ne valittiin.

2.2.1 Viivakoodi

Viivakoodin kehitti Joseph Woodland ja Bernard Silver Yhdysvalloissa vuonna 1949. Woodland ja Silver keksivät käyttää morseaakkosten viivoja ja pisteitä tiedon tallentamiseen. Pisteet olivat kuitenkin liian hankalia luettavaksi, joten he päättivät venyttää pisteet ohuiksi viivoiksi ja viivat paksuiksi viivoiksi. Viivakoodit alkoivat yleistyä 1970-luvulla kun viivakoodinlukulaitteet tulivat markkinoille. [13]

Viivakoodien avulla voidaan tieto koodata optiseen koneellisesti luettavaan muotoon. Viivakoodeissa jokaista merkkiä vastaa mustien ja valkoisten raitojen tai pisteiden yhdistelmä. Aluksi kehiteltiin yksiulotteisia viivakoodeja, joissa tieto on koodattu valkoisiin ja mustiin pystyraitoihin. Koska yksiulotteisilla viivakoodeilla on rajalliset mahdollisuudet esittää tietoa ilman, että niiden pituudet alkavat rajoittaa niiden käyttöä, kehitettiin kaksiulotteisia viivakoodeja. 2D-viivakoodien avulla pystytään sisällyttämään suhteellisen paljon tietoa käytettyyn pinta-alaan nähden.

Viivakoodit luetaan niitä varten tehdyllä lukijalla. Yksinkertaisin lukija muodostuu valonlähteestä ja yhdestä fotodiodista. Tällainen lukija on malliltaan kynänmuotoinen kuten kuvassa 1 oleva. Lukijan toiminta perustuu siihen, että valonlähteestä lähtevä valo heijastuu viivakoodin valkoisista alueista, kun taas mustista viivoista valoa ei heijastu. Tätä takaisin heijastunutta valoa luetaan fotodiodilla. Fotodiodilla syntynyttä virran vaihtelua käytetään viivakoodin viivojen ja tyhjien välien leveyksien mittaamiseen ja täten saadaan viivakoodi luettua. [32,42]

Kuva 1. Kynänmallinen viivakoodinlukija [37]

Kehittyneemmässä lukijassa voidaan käyttää valonlähteenä laservaloa. Kun lukijaan lisätään edestakaisin liikuteltava peili tai pyörivä prisma, saadaan valonlähde ja fotodiodi liikkumaan viivakoodilla ilman, että käyttäjän täytyy itse liikuttaa lukijaa. Tälläinen lukija on malliltaan pistoolityylinen (Kuva 2). Takaisin heijastuvaa valoa luetaan samalla tavalla fotodiodin avulla. Lukija on suunniteltu niin, että se tunnistaa vain lukijan oman valon heijastuksia. [32,42]

Kuva 2. Laser-viivakoodinlukija

CCD-kennollisissa lukijoissa on satoja sensoreita rivissä, jotka mittaavat valon intensetiteettiä suoraan sensorin edestä. Koska sensoreita on satoja rivissä saadaan näiden tuottamasta jännite-erosta muodostettua sähköinen malli viivakoodin mustien raitojen ja tyhjien valkoisten paikkojen määristä ja leveyksistä. CCD-kennollinen lukija eroaa kynä- ja laser-viivakoodinlukijoista sillä, että se ei tarvitse omaa valonlähdettä vaan se voi käyttää hyödyksi ympäröivää valoa. CCD-kennollisen lukijan lukuetäisyys on sen toimintaperiaatteesta johtuen lyhyempi kuin esimerkiksi laser-lukijan. [32,42]

Uusimmat viivakoodinlukijat perustuvat kameraan. Kameran avulla otetaan kuvia, joita käsitellään niin, että niistä pystytään tulkitsemaan luettu viivakoodi. Tällainen viivakoodinlukija voi perustua CCD-kennolliseen videokameraan, korkearesoluutioiseen teollisuuskameraan tai vaikka matkapuhelimen kameraan. [32]

Viivakoodeja on eri muotoisia. Yleisimpiä muotoja on esimerkiksi CODE 39, joka sisältää kirjaimet ja numerot. Siinä jokaista merkkiä esittää 5 viivaa ja 4 väliä. Sarjan

aloittaa aloitusmerkki ja sarjan lopettaa lopetusmerkki [17,43]. CODE 39 viivakoodeja on käytössä esimerkiksi kirjastoissa kirjojen tunnisteissa ja kirjastokorteissa. Kuvassa 3 on esimerkki CODE 39 -muotoisesta viivakoodista.

Kuva 3. Esimerkki CODE39-muotoisesta viivakoodista.

CODE 128 mahdollistaa 128 eri merkkiä. Sitä käytetään yleensä pankkiviivakoodeissa. Myös Kyrel Oy:n työkorttien viivakoodit ovat CODE 128 -muodossa. Koodissa on tarkistussumma, joka lasketaan summaamalla ensimmäinen merkki jokaisen merkin ja niiden paikan tulolla ja sen jälkeen otetaan saadusta luvusta jakojäännös luvun 103 kanssa [7]. Kuvassa 4 esimerkki CODE 128-muotoisesta viivakoodista.

Kuva 4. Esimerkki CODE 128-muotoisesta viivakoodista.

Tunnetuin viivakoodi on kuitenkin EAN-viivakoodi, jota käytetään maailmanlaajuisesti vähittäiskaupassa yleisesti myytyjen tuotteiden tunnuksena. Perusmuotona on EAN-13-viivakoodi joka on 13-merkkinen. Sen rinnalla on käytössä myös EAN-8, jota käytetään lähinnä pienemmillä tuotteilla. Joskus EAN-13-viivakoodin perään lisätään joko EAN-2- tai EAN-5-koodi, joiden pituudet ovat kaksi tai viisi merkkiä. Tällöin saadaan 14- tai 17-merkkinen koodi, joita voidaan käyttää esimerkiksi erottamaan aikakauslehtien eri

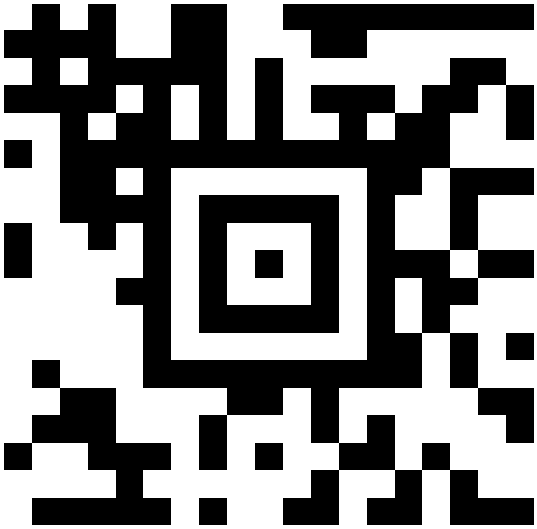
numerot toisistaan tai kaupassa punnittavien elintarvikkeiden ja muiden tuotteiden painon ilmaisemiseen. Kuvassa 5 on esimerkki EAN-koodista. [8,45]

Kuva 5. Esimerkki EAN-muotoisesta viivakoodista.

2.2.2 Kaksiuloitteiset viivakoodit

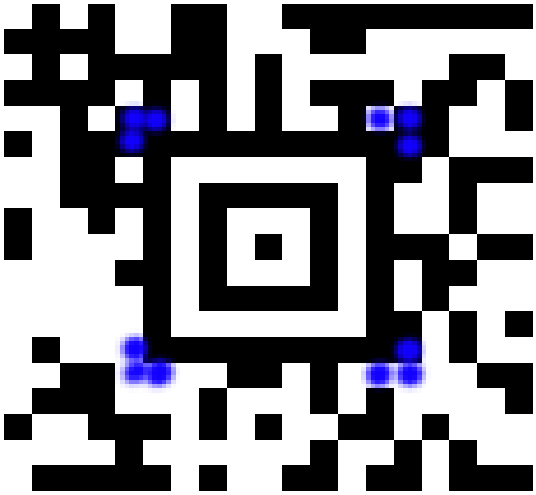
Kaksiulotteisia viivakoodeja on kehitetty useita erilaisia. Toisin kuin yksiulotteiset viivakoodit, kaksiuloitteiset viivakoodit vaativat kuvasensoriin perustuvan lukijan. Ensin kuva luetaan, tämän jälkeen sitä käsitellään digitaalisesti niin, että siitä saadaan luettua sen sisältämät tiedot. Luettu tieto dekoodataan selkokieliseen muotoon. Nykyään tunnetuimmat ja käytetyimmät kaksiuloitteiset viivakoodit ovat DataMatrix ja QR. Seuraavissa kappaleissa on kerrottu yleisimmistä kaksiuloitteisista viivakoodeista.

Aztec Code:n kehitti Andrew Longacre Jr. ja Robert Hussey vuonna 1995. Aztec Code oli alunperin patentoitu, mutta patentista tehtiin myöhemmin vapaa. Aztec Code julkaistiin myös ISO/IEC 24778:2008 standardina. Aztec Coden erikoisuus muihin kaksiuloitteisiin viivakoodeihin verrattuna on se, että se ei tarvitse koodin ympärille hiljaista-aluetta kuten muut koodit mahdollistaen sen mahduttamisen pienemmälle alueelle. Aztec Coden nimi tulee sen keskellä olevasta neliön mallisesta kohdistusalueesta, joka muistuttaa atsteekki-intiaanien pyramideja.
Kuvassa 6 on esimerkki Aztec Codesta. [16,41]



Kuva 6. Esimerkki Aztec Codesta.

Aztec Codessa tieto on koodattu keskikuvion ympärille neliömäisissä kehissä. Uloimman kohdistuskehän jokaisen kulman ympärillä on kolmen pikselin alueet, joita käytetään koodin suunnan määrittämiseen. Kuvassa 7 nämä pikselit on merkattu sinisellä. Niiden luku alkaa kulmasta, missä on kolme mustaa pikseliä, tämän jälkeen kulmaan, missä on kaksi mustaa pikseliä, kolmanneksi kulma, missä on yksi musta pikseli ja lopuksi kulma, missä on vain valkoisia pikseleitä. Näin pystytään määrittelemään aina miten päin koodia pitää lukea. Näiden pikselien väliin jääviä seitsemän pikselin välejä käytetään koodaustietojen säilyttämiseen. Suunnanmääritys- ja koodaustietojen jälkeen alkaa varsinaisen tiedon säilyttämiseen tarkoitetut kehät. [16,41]



Kuva 7. Aztec Coden rakenne.

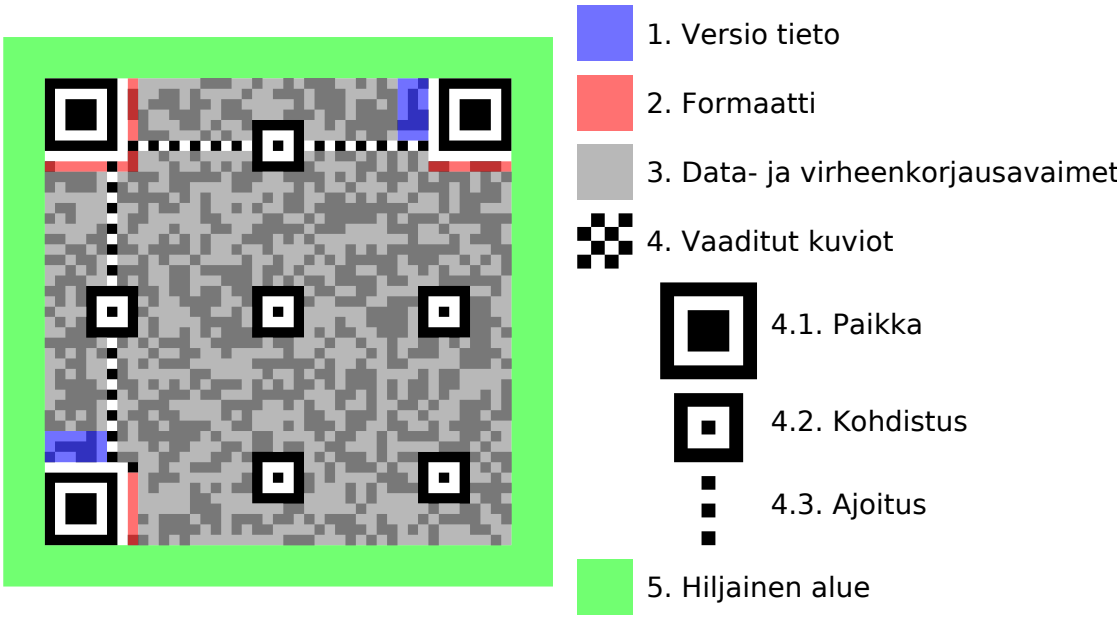
QR-koodi kehitettiin Japanissa Denso Wave -nimisessä yhtiössä vuonna 1994. Nimi on lyhenne sanoista ”Quick Response”, jolla tarkoitetaan koodin nopeaa lukemista ja purkamista. QR-koodi kehitettiin alunperin teollisuuden nopeaksi tuotannonseurantajärjestelmäksi. Nykyään QR-koodit ovat levinneet nopeasti kaikkialle, esimerkiksi tuotepakettien painatuksiin. Tarkoituksena on, että käyttäjä voi matkapuhelimellaan lukea koodin, joka sisältää esimerkiksi URL-osoitteen valmistajan sivuille tai tuote-esittelyvideoon. QR-viivakoodeja lukevia ohjelmia löytyy nykyään jokaiseen yleisimpään älypuhelimeen. QR-koodeja voi luoda ilmaiseksi useilla eri verkkopalveluilla. QR-koodit ovat avoimia, ja niitä voi kuka tahansa käyttää. Ne on myös julkaistu ISO-standardina. Denso Wave kuitenkin omistaa QR-koodien patentit ja ”QR code” on Denso Waven rekisteröity tavaramerkki. Kuvassa 8 on malli QR-koodista. [34,51]

Kuva 8. Esimerkki QR-koodista.

QR-koodi voi sisältää jopa 4296 merkkiä tai 7089 pelkkää numeroa tai 2953 bittiä, riippuen mitä sillä halutaan esittää. QR-koodien koodaus pohjautuu muutamaan standardiin. QR-koodi voidaan jakaa versioihin 1 – 40, jotka eroavat toisistaan tallennettavan tiedon määrällä. QR-koodissa voidaan käyttää neljää eri asteista korjaustasoa. Tieto on tallennettu 8-bittisiin koodisanoihin, joissa käytetään Reed-Solomon virheenkorjausalgoritmejä. Korkeimmalla korjaustasolla voidaan jopa 30% koodisanoista palauttaa. Mitä suurempi virheenkorjaustaso, sitä vähemmän voidaan tietoa tallentaa. Virheenkorjauksen avulla on mahdollista luoda QR-kooditaidetta. QR-koodin keskelle voidaan sisällyttää esimerkiksi kuva yrityksen logosta tai tekstiä. kuvassa 9 on esimerkki tälläisesta. [34,51]

Kuva 9. Esimerkki värikkäästä QR-koodista.

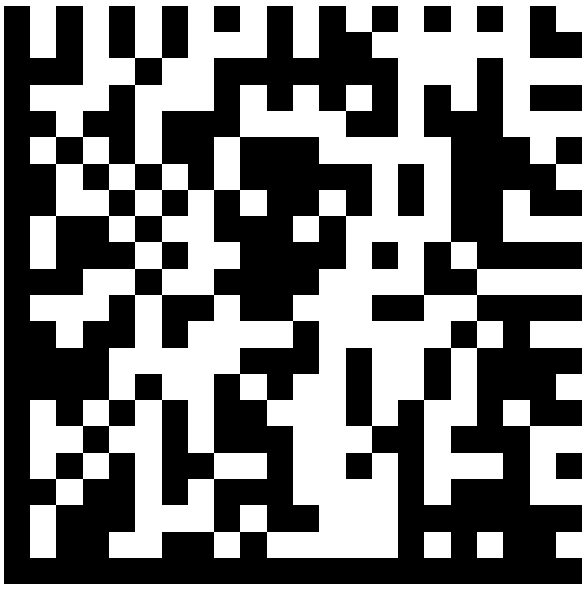
Toisin kuin yksiulotteiset viivakoodit, QR-koodien lukemiseen käytettävän laitteen täytyy perustua kuvasensoriin. Tämän jälkeen kuva tulkitaan ja puretaan digitaalisesti. QR-koodia katsoessa huomaa ensimmäisenä kolme isoa neliötä koodin kulmissa. Näiden avulla määritellään QR-koodin paikka. Pienemmät neliöt, jotka erottuvat koodista, on tarkoitettu kohdistamista varten. Kuvassa 10 näkyvät siniset alueet kertovat käytetyn QR-koodin version ja vihreä alue kertoo QR-koodin vaativan hiljaisen alueen, eli alueen jonka täytyy olla tasavärinen.



Kuva 10. QR-koodin rakenne, perustuu lähteeseen [50]

Data Matrixin kehitti International Matrix Inc (ID Matrix). Nykyään Data Matrix on standartoitu sekä vapaassa että ilmaisessa käytössä useassa eri sovelluksessa. Data Matrixista on tehty useita versioita, joista uusin on ECC 200. Se eroaa edellisistä esimerkiksi virheenkorjauksen osalta. Siinä käytetään virheenkorjaukseen Reed-Solomon korjausta, jota käytetään myös QR-koodeissa. [12,33,44]

Data Matrix voi sisältää numeroita tai tekstiä. Koodiin voi olla tallennettu dataa muutamasta tavusta aina 1556 tavuun asti. Datan määrä riippuu siitä, kuinka monta solua koodissa on. Jokaisessa koodissa on kaksi mustaa reunaa L-muodossa, joiden avulla tunnistetaan Data Matrix -koodin paikka ja sen suunta. Kahta muuta reunaa käytetään ajoitukseen, jonka avulla tiedetään kuinka monta riviä ja saraketta luettavassa koodissa on. Rivien ja sarakkeiden määrä riippuu koodatun tiedon määrästä. Data Matrixia käytetään paljon esimerkiksi teollisuudessa tuotteiden merkitsemiseen. Sitä voidaan käyttää myös pienien esineiden merkitsemiseen, sillä esimerkiksi 2 tai 3 mm^2 koodiin voidaan koodata jopa 50 merkkiä niin, että ne on luettavissa. Kuvassa 11 on esimerkki Data Matrix -koodista. Siitä löytää helposti L-muotoiset kohdistusreunat ja kaksi muuta ajoitusreunaa. [12,33,44]



Kuva 11. Esimerkki Data Matrix-koodista.

2.2.3 RFID (Radio Frequency IDentification)

RFID-pohjaiset ratkaisut ovat kalliimpia kuin viivakoodit, mutta niiden hyötynä on nopeampi luenta, varsinkin useammalla yhtäaikaisella tuotteella. Hyötynä on myös se, että yhtä tunnistetta voidaan uudelleenkäyttää. RFID-tunnisteita on aktiivisia ja passiivisia. Aktiiviset sisältävät virtalähteen, mutta ovat huomattavasti kalliimpia. Tämän takia passiiviset olisivat halvempia hankkia ja käyttää, ja sopisivat paremmin tähän projektiin.

Tunnisteen koko rajoittaa kuitenkin niiden käyttöä pienimmillä tuotteilla. Myös hyvin pieniä tunnisteita on olemassa, mutta niiden hinta ja käsittely saattavat aiheuttaa ongelmia. Pienimmät RFID-tunnisteet, jotka on tarkoitettu piirilevylle, ovat kooltaan 1x35mm, mutta ne vaativat erillisen antennin piirilevyltä. RFID-järjestelmän ongelmana on myös se, että lukijoiden vaikutus valmistettaviin tuotteisiin täytyisi selvittää.

RFID-tekniikoita on erilaisia ja ne toimivat eri taajuusalueilla. Käytetyimmät on HF ja UHF. RFID-tunnisteita löytää nykyään hyvin monenlaisia. Tunnisteita on tehty

esimerkiksi avaimenperän muotoisiksi, luottokorttin malliseksi, eri muotoisiksi ja kokoisiksi tarroiksi, ihon alle tuleviksi implanteiksi ja takin napeiksi. Kuvassa 12 on esimerkkejä RFID-tunnisteista.

Kuva 12. Esimerkkejä RFID-tunnisteista [28]

UHF –taajuusalueella (Ultra High Frequency) käytettävät RFID-taajuudet vaihtelevat maiden välillä, esimerkiksi Euroopassa käytetään 869 MHz taajuutta kun taas Yhdysvalloissa käytetään 902-928 MHz taajuusaluetta. UHF-taajuuksien RFID-järjestelmien lukuetäisyydet ovat pisimmät. Siinä tunniste ja lukija kommunikoivat radioaaltoja käyttämällä. Tämän vuoksi se toimii huonommin nesteiden ja metallien lähellä. Tämän ongelman ratkaisemiseksi on kehitetty ns. lähikenttäUHF -tunnistaminen (Near Field UHF). Se perustuu noin 20 cm etäisyydelle kantavasta tunnistamiseen tarpeeksi vahvasta magneettikentästä. LähikenttäUHF:ää käyttävät järjestelmät toimivat paljon varmemmin metallien ja nesteiden lähellä kuin tavalliset UHF-taajuudelliset järjestelmät. [29]

HF (High Frequency) –taajuusalueella toimivat RFID-järjestelmät käyttävät yleensä 13,56MHz taajuutta. Ne toimivat paremmin metallien ja nesteiden lähellä ja ovat häiriösietoisempia. HF-järjestelmien lukuetäisyydet ovat kuitenkin lyhyemmät, jääden 0.05m ja 1m välille sovelluksesta riippuen. HF-taajuutta käytetään paljon esimerkiksi kulunvalvonnan järjestelmissä. Toisin kuin UHF-taajuuksien kanssa, tapahtuu HF-järjestelmissä tiedonsiirto induktiivisesti, jolloin RFID-tunniste reagoi lukijan luomaan magneettikenttään. [29]

LF (Low Frequency) –taajuusalueella toimivat RFID-järjestelmät toimivat yleensä 125 kHz taajuudella. LF-järjestelmiä käytetään yleisemmin eläintentunnistesiruissa ja joissain kulunvalvonnan järjestelmissä. Uusissa sovelluksissa LF-taajuusaluetta harvemmin nykyään käytetään. [29]

NFC (Near Field Communication) on RFID-tekniikkaa käyttävä lyhyen noin 4cm etäisyydellä toimiva tekniikka. Se on kehitetty matkapuhelinta varten, jolloin matkapuhelin sisältää RFID-lukijan ja tunnisteen. Tämä mahdollistaa kaksisuuntaisen kommunikoinnin laitteiden välillä, jolloin esimerkiksi Bluetooth-kuulokkeet ja matkapuhelin voidaan yhdistää keskenään NFC-kommunikoinnin avulla. [29]

2.2.4 Konenäkö

Konenäköä käytetään nykyään paljon teollisuuden eri työvaiheissa. Sillä pystytään korvaamaan ihminen työvaiheissa, jotka ovat vaikeita, yksinkertaisia, aikaa vieviä, vaarallisia tai mahdottomia ihmiselle. Konenäköä käytetään nykyään usein erilaisissa lajittelu- ja laadunvalvontatehtävissä. Sellaisia työvaiheita on paljon teollisuudessa, mutta konenäköä käytetään paljon myös automaattisissa laitteissa, kuten polttoaine-, seteli- tai pullonpalautusautomaateissa. Konenäkö on myös yleistymässä esimerkiksi parkkihalleissa ja autonpesulaitteissa tunnistamaan auton rekisterikilpiä.

Konenäkö rakentuu useammasta osasta. Ensimmäiseksi on kamera-osa, joka sisältää kuvasensorin ja kameran optiikan. Tämän jälkeen on kuvan muuntaminen ja

tallentaminen digitaaliseen muotoon. Seuraavaksi kuvaa käsitellään halutulla tavalla ja etsitään kuvasta ne ominaisuudet ja asiat joita tarvitaan halutun tehtävän suorittamiseen. Lopuksi kuvasta analysoidaan löydetyt ominaisuudet, joiden perusteella ohjataan koko järjestelmää. [14,40]

Kameran kuvasensorit ovat yleensä joko CCD- tai CMOS-kennoja (Charge coupled device, Complementary metal –oxide semiconductor). Normaalien videokameroiden kuvasensorien kennot muodostuvat pikseleistä, jotka ovat asennettu matriisiksi. Mitä enemmän pikseleitä, sitä tarkempi kuva, jolloin kuvasta voidaan erottaa pienempiä yksityiskohtia. Toisaalta, mitä enemmän pikseleitä, sitä raskaampaa on kuvankäsittely. [14,40]

Konenäkö-järjestelmissä käytetään myös usein viivakameroita. Niissä kuvakenno on viivamainen, jolloin pikselit ovat jonossa. Tällaisia kameroita käytetään usein kuvaamaan esimerkiksi liukuhihnalla liikkuvia osia tai itsensä ympäri pyöriviä esineitä.

Kameran optiikan muuttujia ovat optiikan polttoväli, sulkijan aukon koko, optiset vääristymät ja syvyysterävyysalue. Tärkein tekijä konenäkö-järjestelmissä, kuin myös video- tai valokuvauksessa on valaistus. Konenäkö-sovelluksissa voidaan valaistus toteuttaa usealla eri tavalla. Valaistuksen toteutustapa valitaan sen mukaan, mitä kuvattavasta halutaan esille. Huomioonotettavia seikkoja ovat esimerkiksi valolähteen etäisyys ja sijainti suhteessa kameraan tai kuvattavaan kohteeseen, valon suuntaus ja aallonpituus sekä valon polarisaatio. Konenäkö on kuitenkin liian kallis ja hankala toteuttaa tähän projektiin, joten se jäi pois vaihtoehdoista.

2.3 Välineitä ongelman ratkaisemiseksi

Seuraavissa alaluvuissa käsitellään laitteita, joiden avulla voitaisiin käyttää aikaisemmin esiteltyjä tekniikoita. Koska tuotteita on paljon, ei jokaiseen tuotteeseen pystytä laittamaan omaa tunnistetta, ja tästä johtuen järjestelmässä tultaisiin tarvitsemaan myös käyttäjän syöttämää tietoa. Ensimmäiseksi tutkitaan mahdollisuuksia käyttää RFID- tai

viivakoodinlukijaa yhdistettynä käyttäjän syötteeseen. Luvussa pohditaan myös, miten toteutus onnistuisi Kyrel Oy:ssä. Tämän jälkeen esitellään laitteita, joita pystyisi käyttämään myös seurannan toteutuksessa. Esitellyt laitteet löytyivät internetistä esiselvitystä tehdessä.

2.3.1 RFID-tunniste tai viivakoodi ja käyttäjän syöte

Tuotteiden suuren määrän takia olisi käytännössä mahdotonta olla omaa tunnistetta jokaisella tuotteella. Tämän takia tunniste voisi olla esimerkiksi tuote-eräkohtainen siten, että se kulkisi erän ensimmäisessä tuotteessa, laatikossa tai räkissä. Tunniste voisi liikkua myös jollakin muulla tavoin tuotteiden mukana. Tällä tavoin tuotetieto liikkuu automaattisesti työvaiheesta toiseen, mutta itse ajanmittaus ja kappalemäärien seuranta tapahtuisi työntekijän toimesta esimerkiksi tietokoneella tai tabletilla.

Tunnisteiden pitäisi pysyä joka solussa niin kauan kuin tuotteita on erästä jäljellä, jotta kaikki erän tuotteita tekevät voivat sitä käyttää kirjauksissa. Ongelmaksi tulee, jos osa erästä siirtyy jo seuraavalle solulle, jolloin siirtyneiden kappaleiden mukana pitää myös seurata tunniste. Tämä aiheuttaa sen, että tunnisteita pitäisi olla useita erää kohden. Tunnisteiden pitäisi olla hyvin eroteltavissa toisistaan, etteivät ne sekaantuisi keskenään. Tämän voi ratkaista sillä, että tunnisteessa lukee tarvittavat tiedot tai tunnisteet olisivat niin hyvin kiinni eri tuotteiden mukana, että ne eivät pääsisi sekoamaan muiden tuotteiden tunnisteiden kanssa.

Tuotannossa on käytössä työkortti joidenkin tuotteiden kanssa. Työkortti sisältää tuotteen eränumeron, tuotteen koodin ja nimen, eräkoon ja työvaiheen nimen ja koodin. Eräkoodi, tuotekoodi ja työvaiheen koodi ovat yhdistettynä yhdellä viivakoodilla, joka löytyy työkortista jokaisen työvaiheen yhteydessä. Tätä viivakoodia voisi käyttää hyväksi projektissa. Sen avulla saataisiin luettua tuotetta ja työvaihetta koskevat tiedot, jolloin käyttäjän tarvitsisi vain huolehtia työn aloitus- ja lopetusajan merkitsemisestä, sekä tehtyjen kappalemäärien syöttämisestä. Viivakoodin lukeminen vie kuitenkin aina jonkin verran aikaa. Lukemisen pitäisi siis tapahtua mahdollisimman automaattisesti,

nopeasti ja luotettavasti jottei siitä synny turhaa ajan hukkaa ja ongelmia käyttäjälle.

Työkortin viivakoodin voisi lukea tietokoneeseen kytketyllä viivakoodinlukijalla tai tabletilla liittämällä siihen esimerkiksi Bluetooth- tai USB-viivakoodinlukija. Tavallinen tietokone olisi kuitenkin hankala sijoittaa joka soluun ja se veisi paljon tilaa ja toisi myös kustannuksia. Tilan tarvetta voisi helpottaa käyttämällä pieniä integroituja tietokoneita tai rakentaa sellainen esimerkiksi mini-ITX-emolevyn pohjalta. Mahdollista olisi myös rakentaa laite käyttämällä esimerkiksi pientä Raspberry Pi –tietokonetta.

Yksi mahdollisuus olisi käyttää RFID-tunnisteita, jolloin tunnisteiden lukeminen kävisi nopeammin ja mahdollisimman automaattisesti. RFID-lukijan pystyisi liittämään tietokoneeseen, mutta tällöin olisi sama ongelma tilankäytön suhteen kuin viivakoodinlukijan ja tietokoneen ratkaisussa. Olemassa on myös Bluetoothin avulla toimivia lukijoita, jotka voisi saada toimimaan tabletin kanssa

Erillinen lukija ja päätelaite mahdollistaisivat niiden asentamisen työpisteeseen parhaalla tavalla käytön kannalta, mutta veisivät luultavasti enemmän tilaa. Lukijoiden akun lataamisen huolehtimisesta tulisi ongelma. Tabletin ja lukijan keskenään yhdistämisen kanssa saattaisi tulla ongelmia käyttäjien näkökulmasta. USB-johdolla varustettu lukija taas lisäisi johtoja ja aiheuttaisi rajoitteita laitteiden sijoittamisen suhteen. USB-liitettävän lukijan kanssa voi tulla myös ongelmia laitteiden akkujen latauksien suhteen, koska tableteissa on yleensä vain yksi USB-liitin, jota käytetään myös tabletin lataukseen. Erillinen päätelaite ja lukija mahdollistaisivat esimerkiksi sen, ettei kumpaakin laitetta tarvitsisi vaihtaa uuteen kun toinen hajoaa tai se halutaan päivittää uuteen.

2.3.2 Integroidulla RFID- tai viivakoodinlukijalla varustettu kosketusnäyttölaite

Laitteita, joissa on sisäänrakennettuna RFID- tai viivakoodinlukija on myös olemassa. Tällainen laite on esimerkiksi Idesco Access Touch (kuva 13). Idesco myy laitetta yrityksille, jotka voivat muokata laitetta käyttötarkoituksen mukaisesti. Access Touch on

suunniteltu käytettäväksi esimerkiksi kulunvalvonta-, tunnistus-, seuranta- tai myyntisovelluksissa. Käyttöjärjestelmänä siinä voidaan käyttää joko Windowsia tai Linuxia. [19]

Kuva 13. Idesco Access Touch [18]

Idescon ratkaisun lisäksi on olemassa Android-tabletteja, joihin on integroitu RFID- tai viivakoodinlukija. Tabletit, joissa on integroidut lukijat, on kuitenkin suunniteltu yleensä teollisuuskäyttöön, ja ovat siksi turhan kalliita tätä projektia ajatellen. Kuvassa 14 on esimerkki RFID-lukijan sisältämästä Android-tabletista. NFC:tä tukevat Android-puhelimet ovat yleistymässä, ja onkin mahdollista, että sitä tukevia Android-tablettejakin on tulossa. Tällöin tuotteiden tunnisteina voisi käyttää NFC-tunnisteita ja niiden lukemiseen voisi käyttää tabletteja, jotka olisivat halvempia kuin teollisuuskäyttöön suunnitellut versiot.

Kuva 14. Android-tabletti RFID-lukijalla. [3]

2.4 Ensimmäisen osion yhteenveto

Koska olemassaolevat työkortit ja niiden viivakoodit olivat kustannuksiltaan edullisin vaihtoehto, päätettiin uudessa järjestelmässä käyttää niitä. Ajatus siitä,että tablet voisi olla sopiva pääte muodostui, koska siten lukulaite saataisiin lähelle käyttäjää ilman, että se veisi liikaa tilaa työpöydällä. Koska tableteissa on myös kamera, vältyttäisiin erillisen lukijan tuomista ongelmista käyttämällä sitä viivakoodien lukemiseen.

Integroidut ratkaisut tuntuivat turhan kalliilta, joten projektia päätettiin lähteä toteuttamaan tavallisilla Android-käyttöjärjestelmää käyttävillä tableteilla. RFID-tunnisteisiin perustuva ratkaisu olisi vaatinut erillisen lukijan liittämistä joko

tietokoneeseen tai tablettiin, joten se olisi vaatinut enemmän hankittavia laitteita. Lisäksi olisi pitänyt hankkia RFID-tunnisteita ja keksiä tapa niiden hallintaan ja määritellä työntekijä, joka pitäisi niistä huolta. Konenäköön perustuvat järjestelmät olisivat olleet myös liian arvokkaita ja vaikeita toteuttaa, koska tuotteet eivät liiku tuotantotiloissa niin, että niiden kuvaus olisi onnistunut järkevästi.

Esiraportin jälkeen lähtökohtana uudelle järjestelmälle olivat seuraavat vaatimukset:

- Edullinen
- Luotettava
- Helppokäyttöinen
- Työkortteja käytetään hyväksi
- Pienikokoinen
- Päivitettävissä
- Tavallinen Android-tabletti

3 KÄYTETTÄVYYS

Koska järjestelmää kehitettäessä oli heti alusta asti ollut mielessä tarjota käyttäjälle mahdollisimman yksinkertainen ja helppokäyttöinen käyttöliittymä, oli sovelluksen käytettävyys suunnittelun lähtökohtana. Tämän takia oli syytä perehtyä tapoihin, jolla käytettävyyttä määritellään. Seuraavissa luvuissa käydään läpi käytettävyyden määritelmiä ja kerrotaan, miten käytettävyyttä voidaan mitata. Luvuissa kerrotaan myös laajemmasta käyttäjäkokemuksen käsitteestä.

3.1 Käytettävyyden määritelmä

Yksinkertaistettuna tuotteen tai palvelun käytettävyys tarkoittaa sitä, kuinka helposti tuotetta tai palvelua voi käyttää halutun toiminnon suorittamiseen [46]. Se muodostuu siitä, kuinka hyödyllistä järjestelmää on käyttää ja kuinka helposti sen käyttö onnistuu. Käytettävyys on määritelty myös ISO-9241 standardin 11:sta osassa. Käytettävyys on hyvin tärkeä asia ja se pitäisi ottaa huomioon alusta asti tuotetta tai palvelua kehitettäessä. Alla on listattuna hyvän käytettävyyden tuomia etuja:

- Lisää tuottavuutta
- Käyttäjällä menee vähemmän aikaa sovelluksen käytön opiskeluun
- Tehtävien suorittaminen nopeutuu
- Tulee vähemmän virhetoimintoja ja niistä toipuminen nopeutuu
- Tukihenkilöitä ja koulutusta tarvitaan vähemmän
- Tuotteen laatu paranee
- Käyttäjät ovat tyytyväisempiä

Käytettävyys saadaan paremmaksi käyttäjäkeskeisellä suunnittelulla. Tällöin järjestelmää suunnitellessa otetaan heti aluksi mukaan myös käyttäjiä ja heihin ollaan

vuorovaikutuksessa koko järjestelmän kehityksen ajan. Käyttäjäkeskeisellä suunnittelulla saadaan järjestelmästä käyttäjälle miellyttävä ja tehokas työväline, jolloin tehokkuus kasvaa. Suunnittelussa pitäisi olla mukana ihmisiä laaja-alaisesti, jolloin usean ryhmän tarpeet tulevat esille ja mahdollisimman suuri osa ongelmista saadaan tuotua esille jo aikaisessa vaiheessa.

Nykyään käytettävyyttä on alettu ottaa paremmin huomioon, varsinkin julkiseen käyttöön tarkoitetuissa sovelluksissa. Tämä johtuu siitä, että ihmiset ovat alkaneet arvostamaan käytön helppoutta. Monen palvelun kesken on kova kilpailu käyttäjistä ja se lisää käyttäjäkeskeisyyden tarvetta palveluita suunnitellessa.

Käytettävyyden määritelmiä on tehty useita. Seuraavissa alaluvuissa on kerrottu tärkeimmät määritelmät.

3.1.1 Käytettävyys ISO-9241-11 mukaisesti

ISO-9241 standardi koskee näyttöpäätetyön ergonomiaa ja sen 11:sta osa koskee käytettävyyttä [20,31]. Standardissa käytettävyys määritellään seuraavasti:

”*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.*” [20]

Yleisesti käytettävyydellä tarkoitetaan standardin mukaan siis sitä, kuinka tehokkaasti, taloudellisesti ja miellyttävästi määritellyt käyttäjät voivat käyttää määriteltyä tuotetta määritellyn tehtävän suorittamiseen tietyssä käyttökontekstissa. [38]

Standardin määritelmän mukaan vaikuttavuus (effectiveness) tarkoittaa sitä, kuinka tarkoin ja täydellisesti määritellyt käyttäjät kykenevät saavuttamaan määritellyt tavoitteet tietyssä käyttöympäristössä. Tehokkuus (efficiency) tarkoittaa sitä, kuinka taloudellisesti määritelty tavoite saavutetaan määritellyssä käyttöympäristössä. Miellyttävyys (satisfaction) tarkoittaa sitä, kuinka miellyttäväksi käyttäjä kokee

määritellyn tehtävän suorittamisen määritellyssä ympäristössä. [20,31,38]

3.1.2 Käytettävyys Nielsenin mukaan

Tohtori Jakob Nielsen on tunnettu ja hyvin arvostettu käytettävyyden asiantuntija. Hänen ansiostaan käytettävyyden käsite on tullut tunnetuksi ja hänen vuonna 1993 julkaistuun teokseensa Usability Engineering viitataan lähes jokaisessa käytettävyyttä käsittelevässä artikkelissa. Hän on jakanut käytettävyyden alla lueteltuun viiteen osaan. [22]

Opittavuus (learnability):

- Mittaa kuinka nopeasti käyttäjä, joka ei ole aikaisemmin käyttänyt tarkasteltavaa sovellusta, oppii käyttämään sovellusta määritellyn tason mukaisesti.

Käytön tehokkuus (efficiency):

- Mittaa kuinka nopeasti määritelty henkilö pystyy suoriutumaan annetuista tehtävistä.

Muistettavuus (memorability):

- Mittaa sitä, kuinka nopeasti käyttäjä, joka ei ole käyttänyt sovellusta vähään aikaan, oppii uudelleenkäyttämään sovellusta annetuissa tehtävissä.

Virheet (errors):

- Mitataan, kuinka paljon tapahtuu virheitä käyttäjän käyttäessä sovellusta, sekä määritellään virheiden vakavuus.

Tyytyväisyys (satisfaction):

- Mittaa käyttäjän omaa näkemystä tuotteen käytön mielekkyydestä.

3.2 Käytettävyyden arviointi

Käytettävyyden arviointiin on kehitetty useita erilaisia menetelmiä. Näillä arviointimenetelmillä pystytään löytämään erilaisia käytettävyysongelmia. Nielsen suosittelee käyttämään useampaa arviointimenetelmää aina kun se on mahdollista [23]. Seuraavissa alaluvuissa on lueteltuna eniten käytetyt menetelmät. Yleisesti menetelmät jakautuvat asiantuntija-arvioihin tai menetelmiin, joilla seurataan käyttäjän tekemiä toimintoja. Koska järjestelmän yhtenä vaatimuksena oli sen helppo käyttö, oli syytä paneutua keinoihin, joilla käytettävyyttä voitaisiin mitata.

3.2.1 Heuristinen analyysi

Heuristiikat ovat kokoelma periaatteita tai sääntöjä. Heuristiikkalistoja on tehty useita erilaisia, joista tunnetuimpia ovat Jakob Nielsenin kymmenen säännön lista ja Shneidermanin kahdeksan kultaista sääntöä.

Heuristinen analyysi on helppo, nopea ja edullinen tapa tutkia käyttöliittymän ongelmakohtia. Sitä varten ei tarvita vielä valmista sovellusta, vaan analyysi voidaan suorittaa jo varhaisen vaiheen suunnitelmilla tai prototyypillä. Heuristisessa analyysissa käytetään asiantuntijoita, jotka suorittavat arvion. Suurin hyöty saadaan, kun käytetään 3-4 arvioitsijaa, sillä tätä enemmän arvioitsijoita tuottaa kustannuksia enemmän hyötyyn nähden, ja tätä vähemmällä määrällä ei taas löydetä kaikkia ongelmia. Arvioitsijat käyvät arvioitavaa käyttöliittymää läpi yksin, sillä yhdessä he voisivat vaikuttaa toisen löytöihin ja mielipiteisiin. Parhaan tuloksen saamiseksi arvioitsijan täytyy käydä arvioitava käyttöliittymä läpi useita kertoja, jolloin hän kirjoittaa ylös tekemiään havaintoja. On myös mahdollista käyttää arviointitilanteessa erillistä tarkkailijaa, joka kirjoittaa arvioitsijan huomioita ylös. Arvioitsijoiden tekemistä huomioista kootaan yhteenveto, jonka avulla tehdään tarpeelliset muutokset. Käyttöliittymään tehdyt muutokset voivat kuitenkin aiheuttaa uusia ongelmia ja tämän takia on tärkeää tehdä arviointi kokonaan uusiksi tehtyjen muutoksien jälkeen. [23]

Heuristisen analyysin ongelmiin voidaan lukea se, että arvioitsijan kokemattomuus voi vaikuttaa tuloksiin. Myös löydettyjen ongelmien vakavuus voi olla hankala arvioida. Luotettava arviointi vaatii myös useita eri arviointikertoja, jotta tehtyjen muutoksien tuomat uudet ongelmat löydettäisiin myös.

Usein käytännön projekteissa tehdään omat heuristiikkalistat tai muokataan valmiista säännöistä parhaiten arvioitavaa järjestelmää varten sopiva lista. Tehdyt heuristiset listat sopivatkin hyvin käytettävyyden arvioinnin suuntaviivoiksi tai lähtökohdiksi.

Jakob Nielsenin käytettävyyden kymmenen sääntöä on tunnetuin lista. Moni myöhemmin tehty heuristiikkalista mukailee Nielsenin listan kohtia lisäten tai muuttaen joitakin kohtia. Alla on lueteltuna ja selitettynä jokainen Nielsenin listan kymmenestä kohdasta [21].

Järjestelmän tilan näkyvyys

- Järjestelmän pitää kertoa käyttäjälle missä vaiheessa mennään ja onko järjestelmä ylipäätänsä käytettävissä. Käyttäjän täytyy saada tekemisestään palaute, oli syöte sitten onnistunut tai epäonnistunut.

Järjestelmän ja todellisen maailman yhteys

- Järjestelmän pitää noudattaa käyttäjän kieltä ja noudattaa käyttäjälle tuttuja käsitteitä, termejä ja sanontoja. Järjestelmän täytyy näyttää tieto käyttäjälle loogisessa ja selkeässä muodossa.

Käyttäjän hallinnan ja vapauden tunne

- Järjestelmä ei saa tarpeettomasti rajoittaa käyttäjää. Käyttäjän antamat komennot tulisi pystyä perumaan ja toistamaan. Lisäksi käyttäjä ei saa tuntea olevansa järjestelmän vanki ja järjestelmässä pitäisi olla mahdollisuus päästä nopeasti alkutilaan. Käyttäjän täytyy saada palautetta siitä, mitä on tapahtumassa. Näin käyttäjälle tulee tunne, että hän hallitsee järjestelmää.

Johdonmukaisuus

- Järjestelmässä tulee käyttää asioille samoja käyttäjälle tuttuja nimiä, jotka ovat samanlaisia muiden järjestelmien kanssa. Käyttäjä ei saa joutua miettimään, tarkoittaako kaksi eri sanaa samaa toimintoa eri tilanteissa, tai tarkoittaako kaksi komentoa oikeasti samaa asiaa. Järjestelmässä noudatetaan mahdollisia standardeja ja navigointityyli on samanlainen järjestelmän joka osiossa, esimerkiksi navigointipalkit ja –painikkeet ovat samassa kohtaa koko ajan.

Virheiden estäminen

- Järjestelmässä on selkeä rakenne, jonka avulla käyttäjä on vuorovaikutuksessa sen kanssa. Komennot ovat selkeitä, eivätkä ne muistuta liikaa toisiaan, eikä kahdella eri komennolla saa olla samanlaista toimintoa. Käyttäjältä tulee pyytää varmistus toiminnoille, joita ei voi perua. Järjestelmä tunnistaa mahdolliset virhetilanteet ja estää niiden toistumisen varoittamalla käyttäjää ennen virheen tapahtumista.

Tunnistettavuus ennen muistamista

- Käyttäjälle tulee tarjota selkeät, tunnistettavat vaihtoehdot, koska ihmisen on helpompi muistaa oikea komento rajatusta joukosta kuin muistaa pitkä monimutkainen komento ulkoa. Symboleita ja kuvia voi käyttää muistivinkkeinä. Käyttöliittymän painikkeet ja muut syötteet ovat loogisia.

Käytön joustavuus ja tehokkuus

- Järjestelmän tulee ennakoida ja mukautua käyttäjän tarpeiden muutoksiin, sillä käytön pitää olla joustavaa ja tehokasta sekä aloittelijalle että edistyneelle käyttäjälle. Järjestelmässä pitää olla oikoteitä ja pikakomentoja usein tarvittaviin toimintoihin.

Esteettinen ja minimalistinen suunnittelu

- Laitteen tulee tarjota vain se tieto, mitä sillä hetkellä tarvitaan. Turhaa monimutkaisuutta tulee välttää, eikä käyttäjää saa häiritä turhilla ärsykkeillä. Huomion täytyisi kiinnittyä ensin tärkeimpiin elementteihin.

Virheiden käsittely

- Virheilmoitusten tulee olla selkeitä ja yksinkertaisia käyttäjän omalla kielellä. Ilmoituksen tarkoitus on kertoa ongelma ja mitä käyttäjän tulee tehdä sen korjaamiseksi. Jos mitään tehtävää ei ole, ei ilmoitustakaan kannata näyttää. Virheilmoituksien kuuluisi olla kohteliaita ja lyhyitä.

Opasteet ja dokumentaatio

- Ohjeet ja opasteet tulee olla helposti saatavilla, nopeasti löydettävissä ja mahdollisimman lyhyitä. Järjestelmää tulisi voida käyttää ilman ohjeitakin, vaikka ne usein ovatkin käyttäjälle välttämättömiä. Ohjeita tulisi antaa automaattisesti.

3.2.2 Käytettävyystestaus

Käytettävyystestauksessa seurataan käyttäjän toimintaa kun hän suorittaa annettua tehtävää. Nielsenin ja monen muun käytettävyysasiantuntijan mielestä käytettävyystestaus on tehokkain tapa testata käytettävyyttä. Sen avulla voidaan selvittää millaisissa tehtävissä ja tilanteissa käyttäjille tulee ongelmia ja miten niitä voitaisiin korjata. Käytettävyystestaus vie kuitenkin suhteellisen paljon aikaa ja sen järjestäminen voi olla työlästä. Käytettävyystestauksen tärkeimpiä vaiheita on lueteltuna alla.[6]

Tavoitteiden määrittely

- Aluksi pitää määritellä mitä testillä halutaan saada selville, eli testin tarkoitus.

Aikataulu ja budjetti

- Määritellään missä ajassa ja millaisilla resursseilla testi suoritetaan.

Testiympäristön ja testihenkilöiden valitseminen

- Suunnitellaan ja valitaan missä testi tehdään ja valitaan testihenkilöt.

Testitehtävien laatiminen

- Suunnitellaan ja tehdään testitehtävät ja mahdollisesti testataan ne pilottiryhmällä.

Testauksen suorittaminen

- Suoritetaan itse testi ja kerätään saatu data talteen.

Tulosten analyysi

- Analysoidaan saatuja tietoja.

Raportointi

- Raportoidaan ja puretaan tulokset

Käytettävyystestaus voidaan suorittaa joko tätä varten tehdyssä laboratoriossa tai testattavan tuotteen oikeassa käyttöympäristössä. Testipaikka tulisi valita tapauskohtaisesti, sillä joissain tilanteissa käyttöympäristöllä on suuri merkitys käytettävyyteen ja testin tuloksiin. Testin kannalta tärkeintä on, että testaajat ovat oikeita käyttäjiä. [6]

3.3 Käyttökokemus tai käyttäjäkokemus

Viime aikoina on alettu puhua enemmän käyttäjäkokemuksesta tai käyttökokemuksesta, jolla tarkoitetaan kokonaisvaltaisempaa, käyttäjän tuntemaa kokemusta jostakin tuotteesta tai palvelusta. Hyvä käyttäjäkokemus tarkoittaa sitä, että käyttäjä tuntee olonsa miellyttäväksi ja on tyytyväinen käyttäessään tuotetta tai palvelua. Käyttäjäkokemus on määritelty myös ISO 9241-210 standardissa. Standardi käsittelee vaatimuksia ja suosituksia ihmiskeskeisen suunnittelun periaatteiksi ja toiminta-ajatuksiksi koko tuotteen elinkaaren ajaksi. [47]

Käyttökokemukseen vaikuttaa moni asia, joista keskeisimmät on lueteltuna alla.

Ennakko-odotukset

- Ihmisille muodostuu mainonnan ja viestinnän kautta odotuksia tuotteelle tai palvelulle jo ennen kuin niitä on edes käytettävissä. Ennakko-odotukset vaikuttavat siihen mitä ihmiset tuntevat tuotetta käyttäessään, ja siten myös käyttökokemukseen.

Käytettävyys

- Aikaisemmissa kappaleissa esiteltiin käytettävyyden mittareita ja hyvän käytettävyyden tärkeyttä. Tuotteen käytettävyys vaikuttaa suoraan käyttäjän tunteisiin tuotetta käytettäessä, joten se on hyvin tärkeä osa käyttäjäkokemusta.

Teknologia

- Tuotteen tulisi toimia jokaisessa ajatellussa tilanteessa. Sen suorituskyky tulisi olla mitoitettu siten, että se pystyy palvelemaan käyttäjiä suunnitellulla tavalla esimerkiksi isommalla käyttäjämäärällä tai muuttuvassa ympäristössä.

Jälkimarkkinointi

- Käyttäjäkokemukseen vaikuttaa myös se, kuinka tuotteen tai palvelun tarjoaja suhtautuu käyttäjään tuotteen oston jälkeen. Tällaisia asioita ovat esimerkiksi asiakkaan kannalta mutkaton takuuasioiden hoito, hyvä käytön opastus ja asiakaspalvelu.

4 OMAN RATKAISUN ESITTELY

Oman ratkaisun kehitys lähti liikkeelle miettimällä, mitä tietoja pitää kerätä ja miten ne saisi kerättyä niin, että tietojen avulla pystyttäisiin saamaan selville halutut asiat. Alla on lueteltuna tiedot, joiden arvioitiin riittävän tiedoiksi mitä järjestelmällä pitäisi kerätä.

- Työntekijänumero
- Eränumero
- Tuotenumero
- Työvaihe
- Aloitusaika
- Lopetusaika
- Tehty määrä

Esiraportin perusteella Android-tabletit olisivat tarpeeksi edullisia, jotta niitä pystyttäisiin hankkimaan useita. Ne olisivat myös tarpeeksi pieniä, joten ne mahtuisivat työpöydille. Niiden käyttö olisi myös helppoa ja nopeasti opittavaa kosketusnäytön ansiosta. Lisäksi tablet ei rajoittuisi yhden valmistajan omiin järjestelmiin, joiden tuki ja päivitykset on sidottu kyseiseen valmistajaan. Tabletin käyttöjärjestelmäksi ei oikeastaan ollut muita vaihtoehtoja kuin Android hinnan ja käyttöjärjestelmän avoimuuden takia. Sopivaksi näytön kooksi arvioitiin seitsemän tai kahdeksan tuuman näyttöä. Tämän kokoinen tabletti olisi käytettävyydeltään hyvä, eikä veisi liikaa pöytätilaa. Aluksi testauksessa käytettiin ASUS Memo Pad K00B-mallista Android-tablettia, jossa oli seitsemän tuuman näyttö. Myöhemmin testauksessa käytettiin myös ASUS Memo Pad 8-tablettia jossa oli kahdeksan tuuman näyttö.

Jos tuotantotilassa olisi paljon tablet-tietokoneita, pitäisi niiden päivitys saada tehtyä ongelmattomasti. On hyvin todennäköistä, että järjestelmä vaatisi päivityksiä muuttuvien tilanteiden takia. Koska haluttiin toteuttaa päivitysystävällinen järjestelmä,

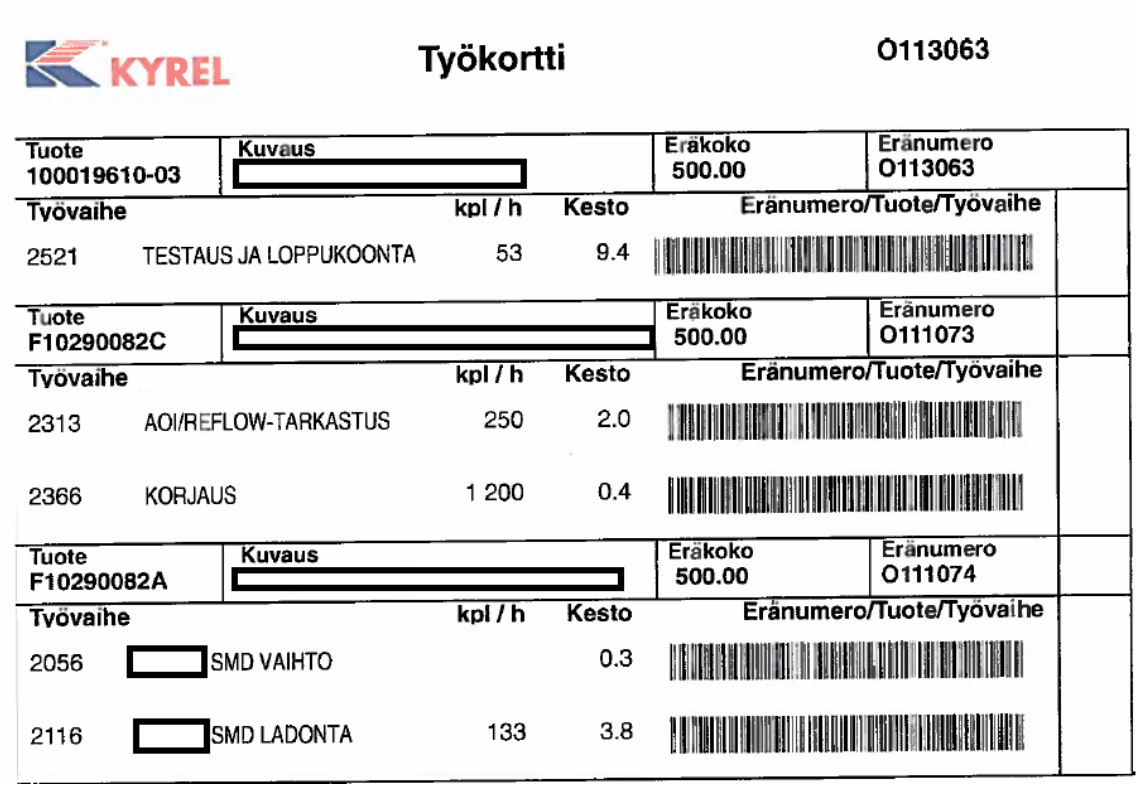
idea web-pohjaisesti sovelluksesta sai alkunsa.

Järjestelmän helppokäyttöisyyteen haluttiin myös panostaa, jotta jokainen työntekijä osaisi käyttää sitä pienellä opastuksella. Järjestelmän pitäisi olla myös niin vikasietoinen, että se kestäisi esimerkiksi verkkoyhteyden katkeamisen.

Näiden määritelmien mukaan järjestelmää lähdettiin kehittämään modulaarisesti niin, että sen muokkaus ja ominaisuuksien lisääminen olisi mahdollisimman helppoa. Käyttöliittymä ja muu koodi eroteltiin toisistaan, ja näiden välillä tieto kulki palvelimelle päin GET- ja POST-pyynnöill. Palvelin palautti tiedon yleensä JSON-muodossa. GET ja POST ovat metodeita millä voidaan lähettää tietoja WWW-sivulta palvelimelle käsiteltäväksi. GET muotoisena tieto välitetään osoitteen mukana ja POST pyynnössä tieto välitetään HTTP-otsikkotietojen mukana. JSON (JavaScript Object Notation) on avoimen standardin tiedostomuoto tiedonvälitykseen.

Koska yrityksessä oli käytössä työkortit, joiden avulla suoritettiin tuotteiden raportointi eri työvaiheiden jälkeen, oli paras idea käyttää työkorttia myös tämän järjestelmän tarpeisiin. Työkortteja ei kuitenkaan tässä vaiheessa ollut käytössä kaikissa tuotteissa, mutta niiden käyttö olisi kuitenkin lisääntymässä ja tarkoitukseni oli, että niitä olisi myöhemmin jokaisella tuotteella. Nämä tiedot vahvistivat ajatuksen niiden käytöstä myös tässä projektissa.

Itse työkortti on kuvan 15 kaltainen. Se sisältää tiedot tuotteesta, siihen kuuluvista rakenteista, työvaiheista ja kyseisen erän tunnuksen. Lisäksi jokaisessa työvaiheessa oli CODE 128:n mukainen viivakoodi, joka sisälsi tunnuksen työvaiheesta, tuotteesta ja erästä. Ajatukseni oli, että sovelluksella pystyisi lukemaan kyseisen viivakoodin, jolloin saataisiin yksinkertaisesti kolme tärkeää tietoa tuotannon seurantaa varten.



KYREL

Työkortti

O113063

Tuote	Kuvaus	Eräkoko	Eränumero
100019610-03	[redacted]	500.00	O113063

Työvaihe		kpl / h	Kesto	Eränumero/Tuote/Työvaihe
2521	TESTAUS JA LOPPUKOONTA	53	9.4	

Tuote	Kuvaus	Eräkoko	Eränumero
F10290082C	[redacted]	500.00	O111073

Työvaihe		kpl / h	Kesto	Eränumero/Tuote/Työvaihe
2313	AOI/REFLOW-TARKASTUS	250	2.0	
2366	KORJAUS	1 200	0.4	

Tuote	Kuvaus	Eräkoko	Eränumero
F10290082A	[redacted]	500.00	O111074

Työvaihe		kpl / h	Kesto	Eränumero/Tuote/Työvaihe
2056	[redacted] SMD VAIHTO		0.3	
2116	[redacted] SMD LADONTA	133	3.8	

Kuva 15. Kuva työkortista.

Ideoita viivakoodin lukemiseen syntyi kaksi kappaletta. Ensimmäinen tapa olisi käyttää erillistä viivakoodinlukijaa USB-liitännän kautta tai langattomasti Bluetoothin avulla. Erillinen lukija olisi kuitenkin nostanut hintaa, vaatinut lisää pöytätilaa ja sen liittäminen tablettiin olisi voinut tuottaa ongelmia. USB-liitännäinen olisi vaatinut, että tabletti tukee OTG-yhteystapaa, jolloin tabletti pystyisi toimimaan USB-isäntänä. Tämä olisi tuottanut ongelmia silloin, kun tablettia olisi pitänyt esimerkiksi ladata USB-portin kautta. Bluetooth-lukijan kanssa olisi tullut ongelmia laitteen yhdistämisessä tablettiin sekä lukijan latauksen huolehtimisesta.

Toinen idea, jolla viivakoodin lukemisen voisi toteuttaa, ja mihin lopulta päädyttiin, oli käyttää tabletin omaa kameraa viivakoodin lukemiseen. Tämän ongelmaksi muodostui se, ettei puhdasta palvelimella pyörivää web-sovellusta voitu käyttää, koska tällä tavoin ei olisi ollut mahdollista hallita esimerkiksi laitteen kameraa.

4.1 Back-end toteutus

Järjestelmä tulisi tarvitsemaan myös palvelimen, jonka kautta saataisiin ladattua näkymien WWW-sivut. Lisäksi tarvittaisiin tietokantapalvelin, johon saataisiin tallennettua töiden tiedot. Aluksi piti valita käytettävät tekniikat. Järjestelmän back-end, eli palvelimella ajettava ohjelma, on toteutettu PHP-komentosarjakielellä. Tähän päädyttiin siksi, että sitä oli mahdollista käyttää olemassa olevalla palvelimella ja siksi, että sitä oltiin käytetty aikaisemminkin.

Tietokantana yrityksessä on käytössä MS SQL, jota käytettiin tässä projektissa. Toiminnanohjausjärjestelmän tietokantana on Pervasive PSQL. Palvelimen käyttöjärjestelmänä on Windows Server 2008 R2. WWW-palvelimena on Internet Information Service 7 (IIS). PHP-koodia suoritetaan IIS:n kanssa käyttämällä FastCGI-rajapintaa. Tietokantaa käytetään Open Database Connectivity (ODBC) kautta, joka toimii rajapintana PHP-koodin ja tietokannan välillä. Seuraavissa alaluvuissa on kerrottu käytetyistä tekniikoista tarkemmin.

4.1.1 Open Database Connectivity (ODBC)

Open Database Connectivity on avoin standardoitu rajapinta tietokannoille. Sen kehitti SQL Access -ryhmä vuonna 1992. ODBC:n kehityksen lähtökohtana oli tarkoitus mahdollistaa kaikkien sovelluksien ja tietokantojen toiminta keskenään. Se toimii rajapintana eri sovellusten ja tietokantojen välillä, ja sen toiminta perustuu ajureihin. Jokainen sovellus ja tietokanta tarvitsee ODBC-ajurin, jonka avulla ne pystyvät keskustelemaan keskenään. [39]

Tässä työssä ODBC:tä käytettiin tietojen lukemiseen toiminnanohjausjärjestelmän tietokannasta ja tämän projektin tietojen tallentamiseen toiseen tietokantaan.

4.1.2 SQL

Structured Query Language (SQL) on alunperin 1970-luvun lopulla IBM:n kehittämä kyselykieli tietokannoille. Nykyään käytännössä kaikki relaatiotietokannat käyttävät SQL-kyselykieltä. SQL:n avulla voidaan tehdä hakuja, lisäyksiä ja muutoksia tietokantaan. SQL ei kuitenkaan määrittele miten komentoja välitetään sovelluksen ja tietokannan välillä. [1]

Microsoft SQL Server (MS SQL) on Microsoftin kehittämä SQL-relaatiotietokanta. Se on laajalti käytössä yrityksissä sen skaalautuvuuden ja tuen takia. MS SQL -serveristä on olemassa useita eri versioita käyttötarkoituksesta riippuen. Pienimmillään yhden tietokoneen pienestä tietokannasta suuriin palvelinkeskusversioihin. [48]

Pervasive PSQL oli alunperin Pervasive Software:n kehittämä tietokanta. Pervasive Softwaren osti Actian Corporation vuonna 2013. Pervasive PSQL on laajalti levinnyt, mutta silti huonosti tunnettu. [26,49]

Tietojen tallentamiseen tässä työssä käytetään MS SQL-tietokantaa, johon luotiin taulu, jonka solut on luetultuna alla.

- ID - tunniste
- employeeID - työntekijänumero
- batch - eränumero
- product - tuotenumero
- stage - työvaihe
- startTime - aloitusaika
- StopTime - lopetusaika
- quantity - määrä
- finish - valmis / kesken

Taulun sisältö vastaa likimain sitä, mitä luvun neljä alussa määriteltiin kerättäviksi tiedoiksi. Ainoastaan ensimmäinen ID-solu toimii tunnisteena ja finish-solua käytetään

työn valmiiksi merkitsemiseen. Jos työ merkataan järjestelmässä valmiiksi, saa tämä solu arvoksi 1, jos työ on kesken, on arvona 0.

4.1.3 Internet Information Services (IIS)

IIS on Microsoftin kehittämä Web-palvelinohjelmisto, joka on tarkoitettu Windows-pohjaisiin palvelimiin. Se on suosittu erityisesti yritysten keskuudessa, joissa lähes kaikki IT-ratkaisut perustuvat Microsoftin tuotteisiin. [25]

Yrityksessä oli käytössä Windows palvelin ja siinä IIS. Näitä ei voinut vaihtaa, vaan ne oli määrätty käytettäväksi tässä projektissa.

4.1.4 Fast Common Gateway Interface (FastCGI)

CGI on standarditapa, jonka avulla luodaan dynaamista sisältöä web-palvelulle. Se toimii rajapintana web-palvelimen ja sen ohjelman välissä, jolla luodaan dynaaminen sisältö WWW-sivulle. FastCGI on jatkokehitetty versio CGI:stä. FastCGI:n avulla ajettava ohjelma on voitu tehdä esimerkiksi PHP-, Python-, Ruby-, C- tai C++ -kielillä. [10]

Kun web-palvelin käynnistää jonkin ohjelman CGI:n avulla, avautuu tästä aina uusi prosessi. Kun näitä prosesseja on useita päällekkäin auki, alkaa palvelimen kyky suorittaa pyyntöjä heikentyä, joka alkaa näkyä palvelun hidastumisena. FastCGI mahdollistaa useamman päällekkäisen pyynnön suorittamisen, koska se ei käynnistä uutta prosessia jokaiselle uudelle nettisivupyynnölle, vaan se avaa pysyvän prosessin, jonka avulla suoritetaan peräkkäin useita pyyntöjä. Tässä projektissa PHP-ohjelmia suoritettiin FastCGI:n kautta. [10]

4.1.5 PHP

PHP on suosittu komentosarjakieli, jonka avulla pystytään tekemään dynaamisia web-sivuja. PHP:n ensimmäinen versio julkaistiin 1995. Sen kehitti tanskalais-grönlantilainen Rasmus Lerdorf. Tällöin PHP oli lyhenne sanoista Personal Home Page Tools. Vuonna 1998 Andi Gutmans ja Zeev Suraski tekivät PHP:stä uuden version, minkä he nimesivät PHP 3:seksi. Tällöin PHP:n uudeksi merkitykseksi valittiin PHP: Hypertext Preprocessor. PHP:llä on laaja luokkakirjasto, joka helpottaa sen käyttöä useissa tilanteissa tarjoamalla valmiin ratkaisun esimerkiksi merkkijonojen käsittelyyn. [15]

Projektin palvelinpuolen ohjelmat toteutettiin PHP:n avulla. Ohjelmasta tehtiin modulaarinen, jolloin ohjelman tarvittavat osat oli pyritty jakamaan omiin tiedostoihin. Tämä mahdollisti helpon laajentamisen ja sovelluksen päivittämisen. Käyttäjälle näytettävät näkymät eivät sisältäneet ollenkaan PHP-koodia, koska nämä asiat haluttiin pitää erillään. Toiminnoissa, joissa palvelin palautti käyttäjälle tietoja, tieto siirrettiin JSON-muodossa. WWW-sivujen lomakkeista tieto siirrettiin GET- tai POST-metodilla PHP-koodille joka käsitteli saadun tiedon. Tällä tavoin saatiin luotua rajapinta toimintojen ja näkymien välille. Rajapintaa apuna käyttäen voitaisiin tehdä uusia ohjelmia eri alustoille. Esimerkiksi voitaisiin tehdä puhelinsovellus jolla selvitetään työntekijänumeron perusteella työntekijän nimi. Tässä voitaisiin käyttää työssä tehtyä checkEmployee.php – tiedostoa joka ottaa GET-metodilla vastaan työntekijänumeron ja hakee tämän jälkeen työntekijän nimen toiminnanohjausjärjestelmän tietokannasta ja palauttaa sen JSON-muodossa.

4.2 Front-end toteutus

Järjestelmän front-end, eli käyttäjän laitteella ajettavat WWW-sivut, joita näytetään näkymissä on tehty käyttäen perinteisiä tekniikoita. Käytössä oli HTML-merkintäkieli, CSS-tyylimäärittely ja JavaScript-komentosarjakieli. Toteutuksessa käytettiin myös HTML 5 -version tuomia ominaisuuksia. Esimerkiksi tehtyjen kappalemäärien

väliaikaiseen tallennukseen käytettiin local storage ominaisuutta. CSS-tyylejä käytettiin esimerkiksi näkymän elementtien aselmointiin ja tekstin koon määrittämiseen. Työntekijän henkilönumeron syöttöön tarkoitetut painikkeet oli myös tehty CSS:n avulla. Työssä käytettiin hyväksi jQuery ja fastClick JavaScript kirjastoja. Jquery on monipuolinen ja suosittu kirjasto, jota käytetään hyvin monessa web-sovelluksessa. Jquery helpottaa JavaScript-koodin kirjoittamista, sillä sen syntaksi on tehty helpoksi [24]. FastClick on kirjasto joka poistaa mobiililaitteiden selaimen 300ms viiveen kun kosketusnäytöllä painetaan painiketta. Mobiililaitteet joutuvat odottamaan noin 300 millisekunttia sen takia, että niiden täytyy pystyä tunnistamaan tuplapainallukset [11]. Tämä odotus saa käyttäjän tuntemaan viivettä painalluksen ja toiminnon välillä, jolloin se alkaa häiritsee käyttöä.

Lomakkeiden tiedot lähetettiin palvelimelle käyttämällä AJAX-tekniikkaa (Asynchronous JavaScript And XML). Niiden avulla saatiin välitettyä tietoja palvelimen ja tabletin näkymien kanssa ilman, että näkymiä olisi pitänyt aina ladata uudestaan. Tällä tavoin saatiin toiminnoista nopeampia ja käyttäjäystävällisempiä.

5 ENSIMMÄINEN VERSIO

5.1 Ensimmäinen prototyyppi

Koska koko projektissa käytettävien ohjelmointikielien ja –tekniikoiden määrä haluttiin pitää vähäisenä, lähdettiin miettimään mahdollisuuksia käyttää JavaScript -komentosarjakieltä myös tabletin laitteiston ohjaamiseen. Tällaisen esimerkiksi JavaScriptin käytön mahdollistavia tekniikoita sanotaan hybridi-sovelluksiksi, koska ne yhdistävät web-teknologioita laitteiston natiivi-sovellusten kanssa, mahdollistaen esimerkiksi kameran käytön sovelluksessa. Tällaisia hybridi-sovelluksen tekemiseen tehtyjä alustoja ovat esimerkiksi PhoneGap ja AppGyver Steroids. Kummankin esimerkin pohjana on Apache Cordova. Kehitysalustaksi valittiin AppGyver Steroids, joka on suomalainen. Steroidsin ominaisuutena on muun muassa nopea tapa testailla sovellusta sen kehitysvaiheessa eri mobiililaitteissa. [4,5,27]

Steroidsia kehitetään aktiivisesti ja sille ilmestyy päivityksiä nopealla tahdilla. Projektin alussa AppGyver Steroidsiin oli jo sisäistetty Cordovan viivakoodinluku-liitännäinen, jota oli tarkoitus käyttää tässä projektissa. Ohjelman prototyypin testauksessa kävi ilmi, ettei viivakoodin luku onnistunut luotettavasti kyseisellä liitännäisellä. Tämän takia käytettäviä tekniikoita alettiin miettiä uudestaan.

5.2 Toinen prototyyppi

Viivakoodin lukeminen ei tuntunut onnistuvan halutulla tavalla AppGyver Steroidsilla tehdyllä sovelluksella, joten seuraavaksi oli tarkoituksena kehittää natiivi-sovellus. Natiivi-sovellus tarkoittaa sitä, että jokaiselle laitealustalle täytyy tehdä oma versio sovelluksesta. Androidin tapauksessa tämä tapahtuisi yleensä Java-kielellä.

Sovelluksessa oli tarkoitus tehdä sellainen, että mahdollisimman moni toiminnoista tapahtuisi palvelimelta ladattavan www-näkymän päällä, jolloin ne olisi helppo päivittää. Tällä tavoin tehtynä täytyisi itse natiivi-osuus ohjelmasta pitää mahdollisimman pienenä, koska sen päivittäminen useampaan laitteeseen olisi lopulta hyvin vaikeaa. Toteutuksessa päädyttiin siihen, että sovelluksen ylälaidan valikkopalkki ja viivakoodin lukemiseen käytetty osuus ovat natiivi-ohjelmaa. Muuten näkymät on ladattu palvelimelta ja ne näytetään Androidin webView:in kautta, jolloin ne olisi aina päivitettyinä. Natiivi-sovelluksen Java-koodi on kirjoitettu käyttämällä apuna Eclipse-ohjelmointiympäristöä ja tämän Android Development Tools -lisäosaa.

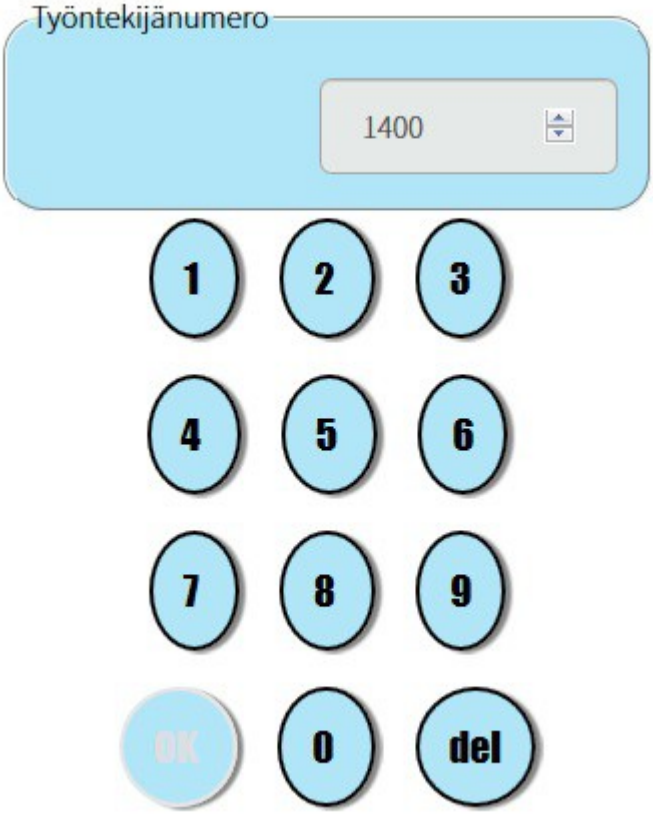
Sovelluksen viivakoodin lukeminen on toteutettu avoimella viivakoodien käsittelyyn tarkoitetulla Zxing Java-kirjastolla. Tämän avulla käytetään tabletin kameraa viivakoodin lukemiseen, ja luettu viivakoodi lähetetään GET-pyynnöllä palvelimelle.

5.3 Näkymät

Sovellus muodostuu näytön yläosan valikkorivistä ja tämän alapuolelle avautuvasta sivukohtaisesta näkymästä. Näkymät ladataan aina palvelimelta Androidin webView-ikkunaan. Viivakoodin lukemiseen tarkoitettu näkymä on kuitenkin Java-pohjainen ohjelma, joten se ajetaan tabletissa itsessään. Seuraavissa alaluvuissa kuvaillaan eri näkymien sisältö ja toiminnot, joita käyttäjä voi kyseisellä näkymällä tehdä.

5.3.1 Työntekijänumeron syöttäminen ja ohjesivu

Sovellus avautuu näkymään, missä kysytään käyttäjän nelinumeroista työntekijänumeroa (kuva 16).



Kuva 16. Työntekijänumeron syöttäminen

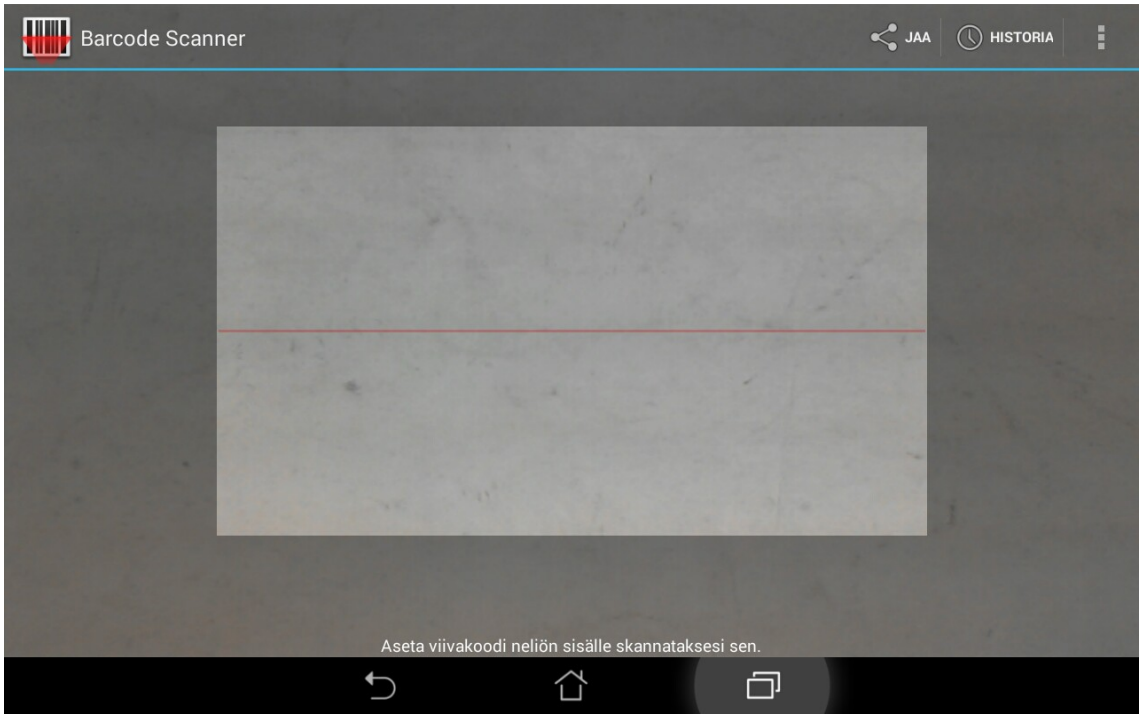
Numeron syötettyään ja OK-nappia painettuaan käyttäjän syöttämä työntekijänumero tarkistetaan toiminnanohjausjärjestelmän tietokannasta. Tarkistaessa varmistetaan, että syötetty numero löytyy, ja tämän jälkeen haetaan tietokannasta työntekijän nimi ja näytetään se käyttäjälle viesti-ikkunassa. Tällä tavoin käyttäjä voi varmistua, että on syöttänyt oikean numeron.

Tämän jälkeen siirrytään näkymään, jossa on lyhyet ohjeet sovelluksen käyttöön. Tätä sivua voidaan käyttää myös ilmoitettaessa muutoksista (kuva 17). Ohjesivun avauduttua käyttäjä voi valita yläreunan painikkeista haluamansa toiminnon.

Kuva 17. Ohjesivu.

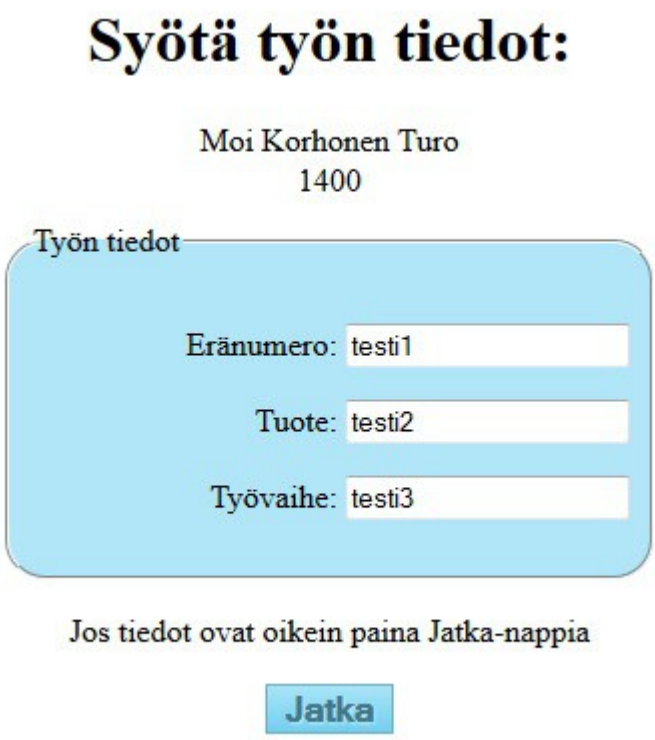
5.3.2 Töiden tietojen syöttäminen ja avoimet työt

Töiden lisääminen tapahtuu painamalla yläpalkista ”lisää uusi työ”-painiketta. Ensimmäisenä avautuu näkymä, jossa saadaan otettua kuva viivakoodista. Tämä näkymä on kuvassa 18. Käyttäjän on tarkoitus kohdistaa tabletin kamera niin, että luettava viivakoodi on punaisen viivan kohdalla. Kun ohjelma tunnistaa viivakoodin kuvasta, se ilmoittaa käyttäjälle mitä koodissa on.



Kuva 18. Viivakoodin lukeminen.

Kun viivakoodi on luettu ja tunnistettu, luettu tieto viedään GET-pyynnöllä palvelimelle. Palvelin käsittelee syötetyn tiedon ja jakaa sen niin, että web-lomakkeen tuote-, työvaihe- ja eränumero-kenttiin saadaan valmiiksi luetut tiedot. Kuvassa 19 on esimerkki näkymästä, johon on tuotu tiedot viivakoodista.



Kuva 19. Työntietojen syöttäminen.

Avoimien töiden näkymässä käyttäjä voi tarkkailla avoinna olevia töitään. Sivulla näkyy taulukossa erän, tuotteen ja työvaiheen tiedot sekä aloitusaika. Näiden lisäksi näkyy kappalemäärälaskuri, kerroin, lisäys-, vähennys-, tallenna- sekä lopeta-nappi. Kerroin-napilla valitaan kuinka suurilla askelilla kappalemäärälaskuri muuttuu lisäys- ja vähennyspainikkeilla. Tallenna-napilla syötetty määrä tallennetaan tietokantaan. Lopeta-painikkeella kappalemäärä tallennetaan ja työ merkitään tehdyksi. Kuvassa 20 on esimerkki avoimien töiden näkymästä. Jos avoimia töitä on useampi näytetään ne allekkain omilla riveillään.

Erä	Tuote	Työvaihe	Aloitusaika	Määrä	Lisää	Vähennä	Tallenna	Lopeta
O503225	109-10-47-D	2321	2015-05-05 12:23:51.000	0	+	-	Tallenna	Lopeta

Korhonen Turo

Kuva 20. Avoimien töiden näkymä.

Kappalemäärät pidetään tallessa paikallisesti laitteella HTML5 local storagen avulla. Local storage tallentaa käyttäjän selaimeen avain/arvo-pareja, joita voi käyttää nopeasti. Tallennetut tiedot pysyvät tallessa, kunnes käyttäjä poistaa ne tai kun sovelluksella halutaan poistaa tietty avain/arvo-pari.

Jos halutaan, että kappalemäärä pysyy tallessa siten, että se on käytettävissä myös muilla laitteilla, voidaan painaa tallenna-nappia. Tällöin kappalemäärä tallennetaan tietokantaan. Jos tietokantaan on tallennettu kappalemäärä kyseiselle erä-tuote-työvaihekombinaatiolle, näytetään kappalemäärässä tietokannasta luettu arvo, jos se on isompi kuin paikallisesti on tallennettu. Jos kappalemäärää ei ole määritelty tietokannassa, näytetään arvona paikallisesti tallennettua arvoa. Käyttämällä local storagea haluttiin varmistua siitä, että sovellusta pystyy käyttämään, vaikka verkkoyhteys olisikin välillä poissa käytöstä tai verkko olisi muuten liian hidas sujuvaan käyttöön. Lopeta-painiketta painettaessa tallennetaan kyseisen työn tietokannan tietoihin lopetusaika.

5.4 Ensimmäisen version testaus

Järjestelmän testaus tapahtui vuoden 2014 kesä- ja heinäkuussa lakkaamossa. Testausaika venyi näin pitkäksi kahdesta syystä. Ensimmäinen syy oli se, että testaukseen valikoitiin vain kolme eri tuotetta, joiden työkortteja luettiin järjestelmään. Toinen syy oli se, että kesällä oli paljon lomia. Heti alussa todettiin, että viivakoodien lukemisen kanssa on ongelmia.

5.4.1 Testauksen havainnot

Kokeilujakson aikana viivakoodien luettavuutta testattiin eri valaistusolosuhteissa sekä testattiin ohjelman erilaisia asetuksia. Viivakoodin lukemiseen käytetyllä paikalla ja täten valaistuksella oli vaikutusta lukujen onnistumiseen, mutta silti viivakoodin lukemiseen meni liikaa aikaa, eikä se ollut tarpeeksi luotettavaa.

Testausta varten tehtiin kyselylomake, jonka järjestelmäntestaukseen osallistuvia pyydettiin täyttämään. Lomake on liitteenä A. Lomake ei kuitenkaan ollut kovin tarpeellinen, sillä testaajien kanssa keskusteltiin testin aikana ilmenneistä ongelmista. Näiden keskustelujen johdosta sovelluksen toimintaa pystyttiin parantelemaan heti. Sovelluksesta pyrittiin tekemään selkeä, helposti opittava ja tarpeeksi yksinkertainen, jotta jokainen pystyisi sitä käyttämään. Testauksessa pystyikin toteamaan näiden seikkojen täyttyvän, ja itse sovelluksen käytön kanssa ei ollut ongelmia.

Testauksessa tuli esille pari virhettä, jotka korjattiin heti. Toinen liittyi tallennettavan kappalemäärän tarkistukseen. Sen avulla tarkistettiin, ettei työtä lopetettaessa olisi syötetty isompaa määrää kuin mitä erän koko oli. Tässä tarkistuksessa oli ehtona < , jolloin järjestelmä ei hyväksynyt koko erän raportointia. Tämä oli helppo korjata lisäämällä ehtoon yhtäsuuruusmerkin, jolloin ehtona oli <=. Tällöin järjestelmä hyväksyi erän kokoa pienemmän tai yhtäsuuren luvun.

Toinen ongelma syntyi, koska järjestelmässä oli testaus, jolla tarkistettiin, että raportoitava erä ja tuote oli oikeasti olemassa. Lisäksi samalla tarkistettiin, ettei tuotetta ole jo raportoitu valmiiksi. Raportoitu-tarkistus jouduttiin kuitenkin poistamaan, koska tietyillä tuotteilla oli samalla tuotekoodilla työvaiheita, jotka oli jo raportoitu tuotannonohjausjärjestelmään ennen lakkaamoa.

Niin kuin aikaisemmin todettiin, oli viivakoodien lukemisen kanssa ongelmia. Viivakoodien lukeminen oli hyvin epäluotettavaa, koska niiden lukemisyritykset muodostuivat pitkiksi ja välillä niiden lukeminen ei onnistunut lainkaan. Välillä lukeminen näennäisesti onnistui, mutta luettu tieto ei ollut oikeanlaista. Lukemisen

aikana tabletin kameran tarkennus seilasi, eikä se yleensä onnistunut tarkentamaan kameraa niin, että tabletti olisi onnistunut lukemaan viivakoodin. Käytetyn tabletin kameran tarkennuksen lukitseminen kiinteäksi olisi taas vaatinut kohteen asettamista mahdottoman kauas tabletista, jotta viivakoodi olisi tullut tarkaksi.

5.4.2 Testaus muilla laitteilla

Ohjelman toimintaa testattiin myös LG G2-puhelimella. Sillä viivakoodien lukeminen onnistui hyvin ja nopeasti. Tästä oli helposti pääteltävissä, että laitteistoissa on eroja. Seuraavaksi oli tarkoitus selvittää, löytyisikö sellaisia Android-tabletteja, joilla viivakoodien lukeminen onnistuisi luotettavasti.

Tablettien testaus suoritettiin käymällä sellaisessa alan liikkeessä, missä oli mahdollisimman paljon eri tabletteja esillä. Liikkeessä käydessä mukana oli työkortin viivakoodi, jolla tabletteja pystyi testaamaan. Testauksen aikana kävi kuitenkin nopeasti selville, että mikään kaupassa esillä olleista tableteista ei pystynyt lukemaan viivakoodeja tarpeeksi nopeasti ja halutulla varmuudella. Muutama malli osoittautui toisia paremmaksi, mutta yllättäen esimerkiksi Samsungin mitkään mallit, eivät edes 400-500 euroa maksavat, pystyneet lukemaan viivakoodia. Viivakoodin lukemista testattiin muun muassa Samsung Galaxy Tab 4, Samsung Galaxy Tab S, HP Slate7 HD, Asus Fonepad ja Acer Iconia tableteilla.

Vaikka jotkut mallit pystyivät viivakoodeja lukemaan, niidenkään toiminta ei ollut tarpeeksi luotettavaa tarkoitettua käyttöä varten.

6 TOINEN VERSIO

6.1 Työkortin jatkokehitys

Ensimmäisen testauksen aikana syntyneiden kokemuksien sekä testattujen tablettien sopimattomuuden johdosta alettiin miettiä vaihtoehtoisia tapoja tuotteiden tietojen luentaan. Koska ennestään oli tiedossa, että Android-puhelimilla ja -tableteilla pystyi lukemaan QR-viivakoodeja hyvin, syntyi ajatus siitä, että niitä voitaisiin käyttää hyödyksi. Tällöin täytyisi työkortteihin saada QR-koodit ja niihin samat tiedot, mitkä olivat CODE 128-muotoisissa viivakoodeissa. Tästä asiasta keskustellessa saatiin päätös niiden käyttöönotosta.

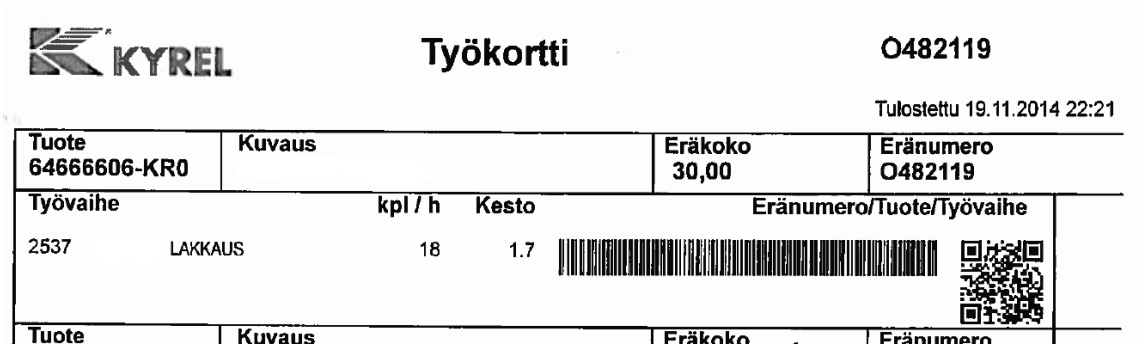
Työkortit tulostetaan Crystal Reports-ohjelman avulla. Se on ohjelma, jonka avulla voidaan luoda raportteja monesta eri data-lähteestä [30]. Itse työkortit muodostetaan toiminnanohjausjärjestelmässä, josta ne siirretään automaattisesti Crystal Reports-ohjelmaan, joka luo niistä tulostettavan sivun. Crystal Reportsiin oli saatavilla liitännäinen QR-koodeja varten. Tämän liitännäisen hankkimisen jälkeen oli mahdollista lisätä QR-koodit työkortteihin.

Kun QR-koodit oli lisätty työkortteihin huomattiin niiden kanssa ongelma. Viivakoodit eivät tulostuneet oikein, vaan ylimmäisen rivin pikselit olivat parin pikselin verran liian oikealla. Tämä aiheutti sen, ettei QR-koodi ollut standardin mukainen eikä sen lukeminen onnistunut. Ongelma ratkesi kun ohjelmistontoimittaja teki järjestelmään tarvittavat korjaukset.

Tämän asian ratkettua huomattiin jälleen uusi ongelma. Joistain työkorteista jäi osa QR-koodeista tulostumatta. Tämä näytti johtuvan siitä, että osa QR-koodeista ei mahtunut niille varatulle alueelle kuten ne oli Crystal Reportsiin aseteltu. Tämä aiheutti sen, että Crystal Reports jätti osan koodeista tulostamatta. Vaikka tämä ongelma oli vielä

olemassa, pystyi QR-koodeja työkorteissa testaamaan. Testaamalla pystyi päättelemään miten työkortteja täytyisi vielä muokata, jotta sen käyttö olisi helppoa.

Ensimmäisessä versiossa QR-koodi oli työkorteissa työvaiherivien viimeisenä 1D-viivakoodien jälkeen (Kuva 21).


KYREL Työkortti O482119

Tulostettu 19.11.2014 22:21

Tuote	Kuvaus	Eräkoko	Eränumero
64666606-KR0		30,00	O482119

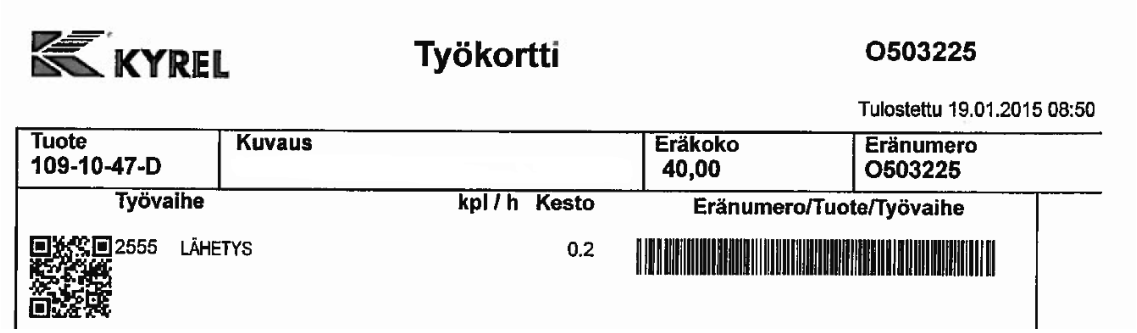
Työvaihe		kpl / h	Kesto	Eränumero/Tuote/Työvaihe
2537	LAKKAUS	18	1.7	

Tuote	Kuvaus	Eräkoko	Eränumero

Kuva 21. Ensimmäinen versio QR-koodin sisältävästä työkortista.

Koska QR-koodien luku onnistui tabletilla vähän liiankin nopeasti oli ongelmana se, että tabletti ehti lukea eri koodin kuin käyttäjällä oli tarkoitus. Koska QR-koodi oli rivin viimeisenä, ja työvaiheen kuvaus rivin alkupäässä, oli käyttäjän hankala saada tabletin kameraa tähdättyä oikeaan QR-koodiin. Tämä johtui siitä, ettei käyttäjä pystynyt näkemään tabletin näytöltä kuvausvaiheessa yhtä aikaa työvaiheen kuvausta ja QR-koodia.

Ongelman ratkaisemiseksi päätettiin QR-koodit siirtää rivin ensimmäiseksi. Tällöin tabletin näytöltä näki kuvausvaiheessa yhtä aikaa QR-koodin sekä työvaiheen kuvauksen. Tämä helpotti huomattavasti oikean viivakoodin löytämistä ja sen lukemista. Viivakoodeja myös pienennettiin, jolloin kaikki QR-koodit mahtuivat työkortille. Kun viivakoodeja pienennettiin, myös niiden luku tabletilla hidastui vähän. Tämä hidastuminen ei kuitenkaan haitannut käyttöä, vaan oikeastaan helpotti käyttäjän työtä, koska käyttäjälle oli helpompaa valita oikea QR-koodi pitemmän lukuajan johdosta. Kuvassa 22 on työkortin viimeisin malli.



KYREL Työkortti O503225

Tulostettu 19.01.2015 08:50

Tuote	Kuvaus	Eräkoko	Eränumero
109-10-47-D		40,00	O503225

Työvaihe	kpl / h	Kesto	Eränumero/Tuote/Työvaihe
2555 LÄHETYS		0.2	

Kuva 22. Viimeisin versio työkortista joka sisältää QR-koodin.

6.2 Sovelluksen toinen versio

Järjestelmän toisen version lähtökohtana oli parannella ensimmäisen version testauksen aikana esille tulleita ongelmakohtia. Lisäksi muutoksia tehtiin toimintaperiaatteeseen.

Uudessa versiossa jätettiin tehtyjen kappaleiden syöttäminen pois, sillä tarkoituksena oli saada kappalemäärä toiminnanohjausjärjestelmästä. Ideana oli, että kun toiminnanohjausjärjestelmässä aloitetaan uusi erä, niin erää luodessa määritellään erään joka tapauksessa tuotteiden kappalemäärä, jota voisi käyttää hyödyksi. Kappalelaskurin + ja – - painikkeet ja kerroinvalikko jätettiin paikoilleen, mutta niiden käyttö oli vapaaehtoista ja niiden tarkoitus oli tarjota käyttäjälle vaihtoehtoinen tapa pitää itselleen kirjaa tehdyistä kappaleista. Uusi toimintaperiaate vähensi työntekijältä vaadittavia toimia ja nopeutti töiden raportointia.

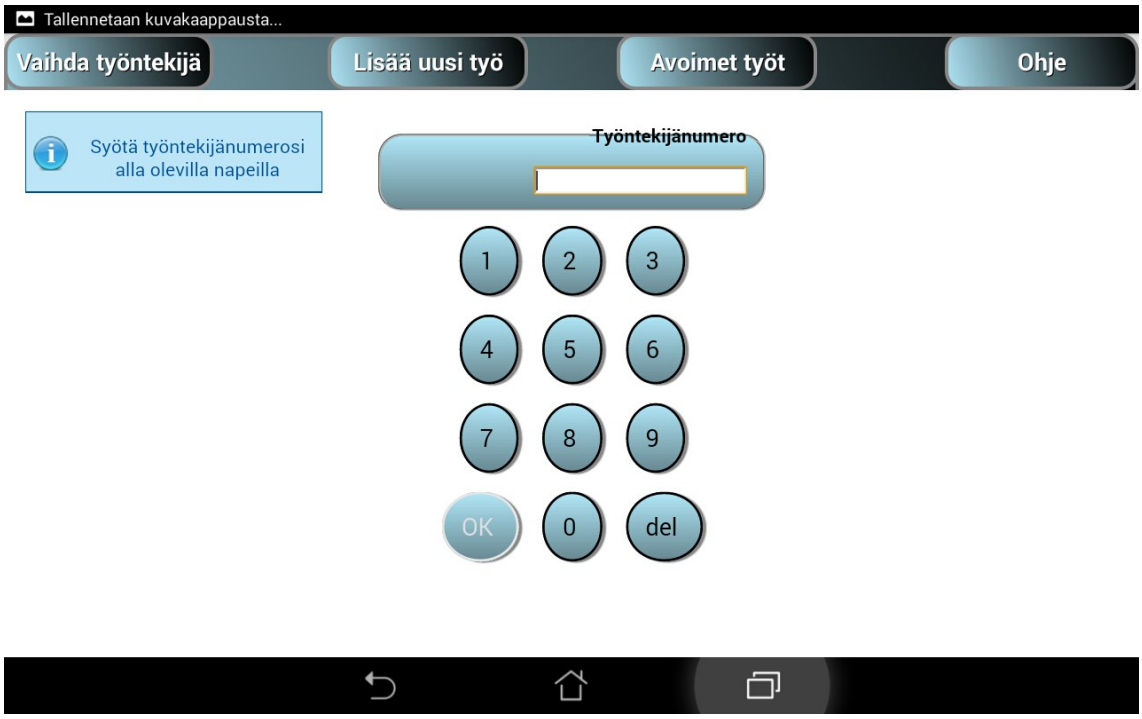
Uuteen tapaan päädyttiin siitä syystä, että haluttiin ratkaista ongelma työaikojenlaskennasta tilanteissa, joissa samaa työvaihetta tekisi yhdessä monta työntekijää. Tällöin olisi muodostunut ongelma siitä, että kuka ja kuinka moni tehdyt kappalemäärät raportoisi järjestelmään. Uudella tavalla laskettiin kaikki tietyn erän ja tietyn työvaiheen kaikki ajat yhteen, ja jaettiin erän kappalemäärä saadulla ajalla,

jolloin päästiin tarpeeksi tarkkaan arvioon yhden kappaleen valmistamiseen kuluneeseen aikaan tietyssä työvaiheessa.

6.3 Näkymät

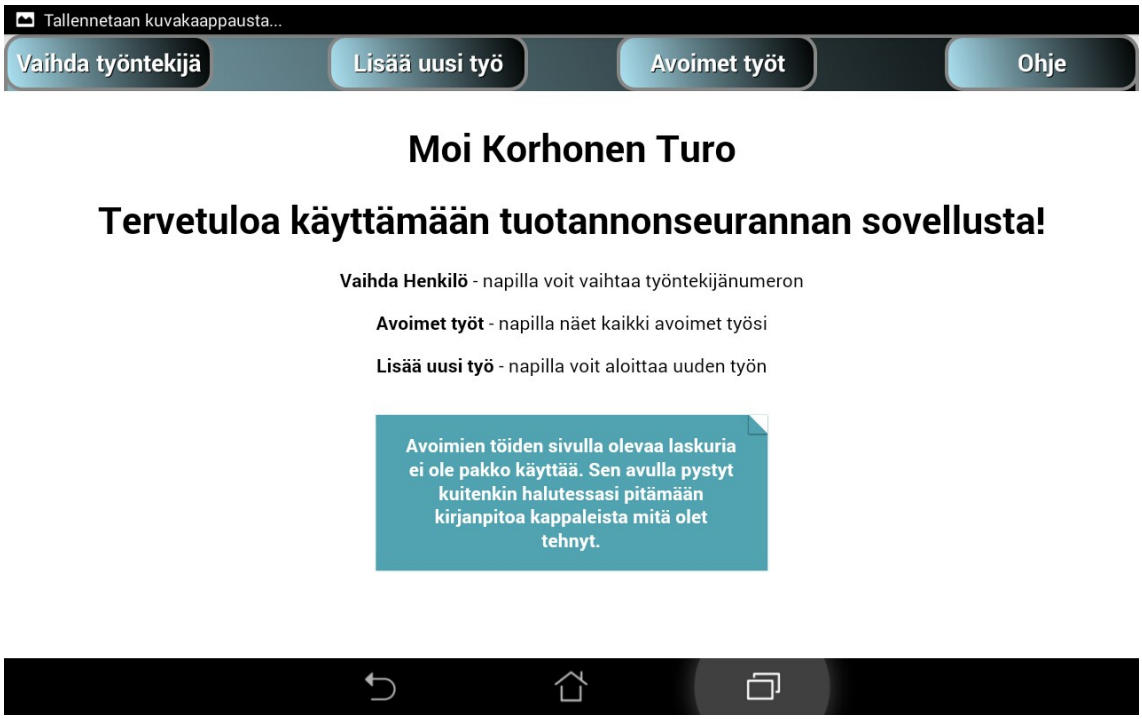
6.3.1 Työntekijänumeron syöttäminen ja ohjesivu

Kuvassa 23 on ensimmäinen näkymä, jonka käyttäjä näkee kun hän käynnistää sovelluksen. Näkymässä on viimeisin versio työntekijänumeron syöttämiseksi.



Kuva 23. Viimeisin versio työntekijänumeron syöttämisestä.

OK-painike aktivoituu vasta, kun käyttäjä on syöttänyt neljä numeroa. Jos syötetty numero on virheellinen, ilmoitetaan tästä käyttäjälle virheilmoituksella ja pyydetään syöttämään työntekijänumero uudestaan. Onnistuneen työntekijänumeron syöttämisen jälkeen avautuu ohje-näkymä (kuva 24).



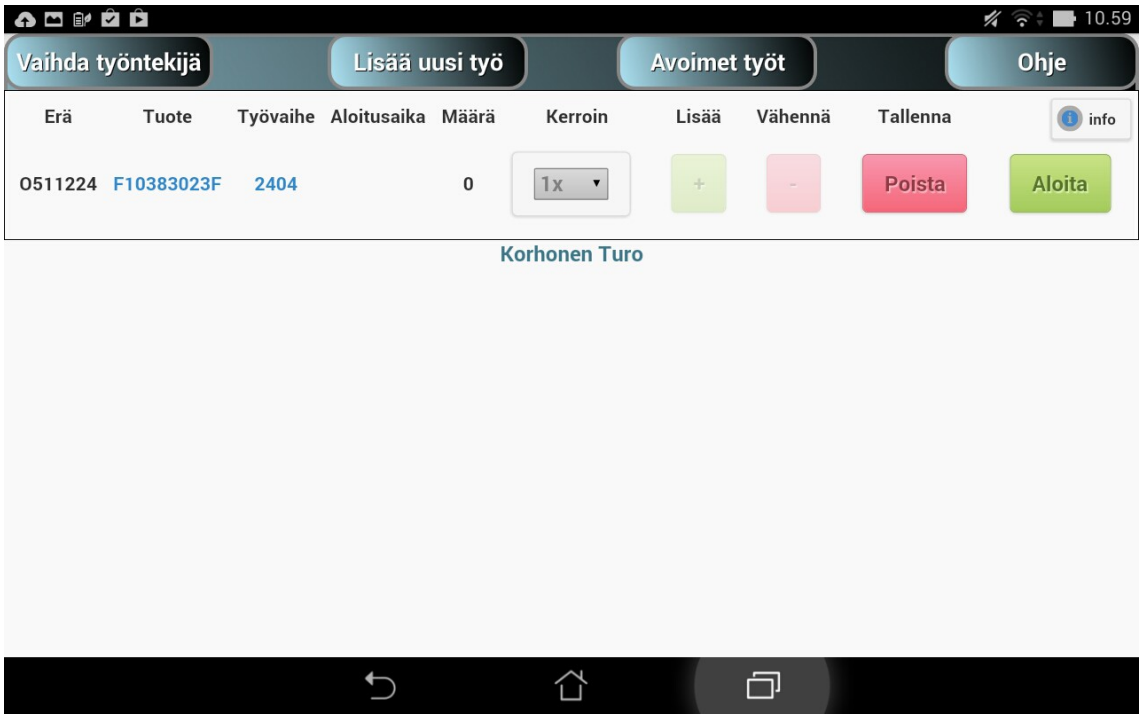
Kuva 24. Viimeisin versio ohje-näkymästä.

6.3.2 Töiden tietojen syöttäminen ja avoimet työt

Kun työntekijä on päässyt ohje-sivulle, voi hän esimerkiksi Lisää uusi työ – painikkeella aloittaa uuden työn lisäämisen. Aluksi avautuu näkymä viivakoodin lukemiseksi (kuva 18). Tämän jälkeen luetut tiedot lähetetään palvelimella ja ladataan näkymä, jossa tiedot voi tarkistaa ja lisätä työn avoimiin töihin. (kuva 25). Uusimmassa versiossa haettiin tähän näkymään tuotteen ja työvaiheen kuvaus, jotta käyttäjä näki heti, minkä tuotteen ja työvaiheen viivakoodi oli luettu. Myös käyttäjän nimeä ja työntekijänumeroa pidetään näkyvissä joka näkymässä, jotta sovellusta käyttävä tietää aina, kenen työntekijänumerolla sovelluksessa ollaan kirjautuneena.

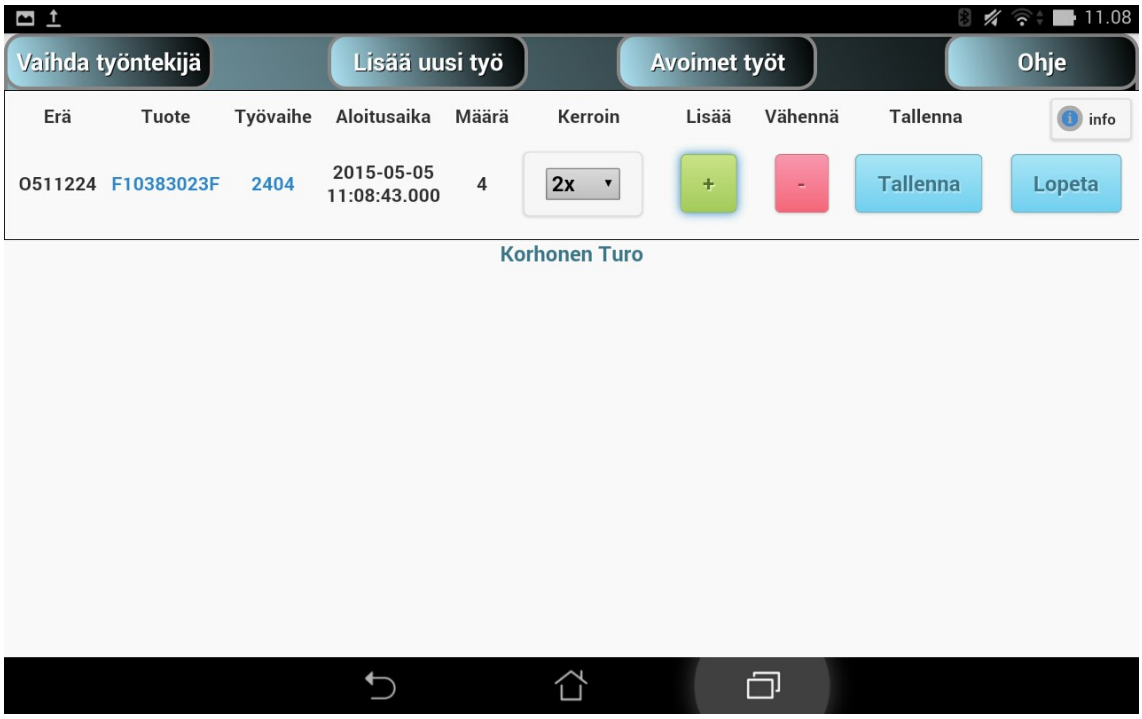
Kuva 25. Työn lisääminen.

Kun käyttäjä on painanut Jatka-painiketta, avautuu näkymä käyttäjän lisäämistä töistä (kuva 26).



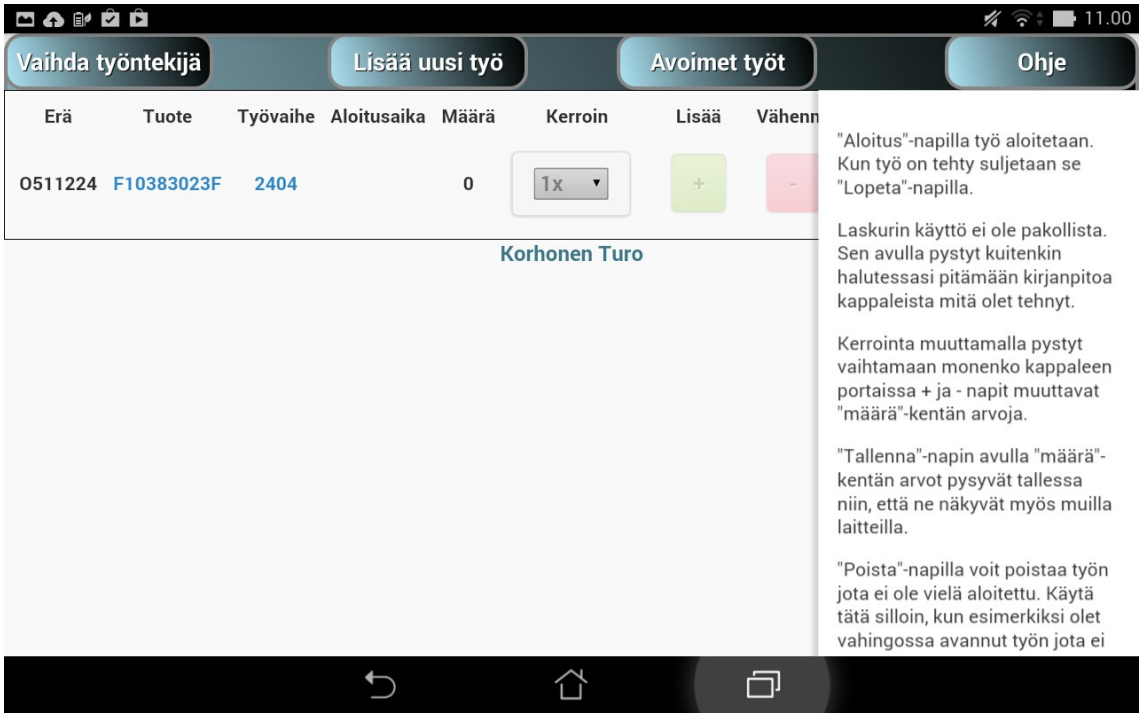
Kuva 26. Näkymä töistä.

Tässä vaiheessa työn tiedot on lisätty tietokantaan, mutta työtä ei ole vielä aloitettu. Kun käyttäjä haluaa aloittaa työn, voi hän painaa Aloita-painiketta. Tällöin työlle tallennetaan aloitusaika tietokantaan ja avoimien töiden näkymä muuttuu kuvan 27 näköiseksi. Alunperin työlle merkattiin aloitusaika heti kun se oli lisätty avoimiin töihin, mutta se muutettiin uudenlaiseksi työntekijöiden palautteesta. Tällä uudella tavalla käyttäjä voi lukea useamman erän ja tuotteen tiedot peräkkäin jo valmiiksi, jolloin on helpompi ja nopeampi vaihtaa erien välillä töitä tehdessä, kun ei tarvitse lukea viivakoodeja erän vaihdon yhteydessä. Turhaan luetun työn voi myös poistaa Poista-painikkeesta.



Kuva 27. Avoimien töiden näkymä, kun työ on aloitettu

Työn aloituksen jälkeen muuttuu +, - ja kerroin -painikkeet aktiivisiksi ja Poista- ja Aloita-painikkeiden tilalle tulevat Tallenna- ja Lopeta-painikkeet. Tallenna-painikkeella työntekijä voi tallentaa kappalemäärän tietokantaan aikaisemmin kerrotulla tavalla. Lopeta-painiketta painaessa enää ei kuitenkaan kysytä kappalemäärää, vaan varmistetaan, että työntekijä haluaa oikeasti lopettaa työn. Näkymän oikeassa laidassa on info-painike, jolla avautuu ohje näkyville (kuva 28).



Kuva 28. Avoimien töiden näkymä ohjeen ollessa näkyvissä.

Painamalla tuotteen koodia avautuu ikkuna, jossa lukee tuotteen nimi. Tämä auttaa siten, ettei käyttäjän tarvitse muistaa tuotteiden koodeja ulkoa. Työvaihetta painamalla aukeaa ikkuna, jossa on kuvaus työvaiheesta.

6.4 Toisen version testaus

Järjestelmän toisen vaiheen testaus aloitettiin tammikuussa 2015 yhdessä tuotannon solussa. Solussa oli käytetty aikaisemmin työaikojen kirjaukseen tietokoneella tapahtuvaa työaikojen syöttöä, joten heiltä saisi palautetta järjestelmien keskinäisistä eroavaisuuksista. Ennen testauksen aloittamista suoritettiin tablet-sovelluksen käytettävyyden arvio Nielsenin kymmenen säännön heuristiikkalistan avulla. Tämä arvion huomiot löytyvät luvusta 6.4.1. Luvussa 6.4.2 on kerrottu testauksen aikana syntyneitä huomioita.

6.4.1 Sovelluksen käytettävyyden arviointi

Jo ensimmäisen version testauksen yhteydessä suoritettiin käytettävyystesti havainnoimalla käyttäjien toimia ja suorittamalla haastatteluja testauksen aikana. Viimeisimmän version testauksen aikana suoritettiin myös haastatteluja. Osa liitteessä A olevan kyselylomakkeen kysymyksistä koskivat käytettävyyttä. Sovelluksen viimeisimmän version pohjalta suoritettiin arviointi käyttämällä Nielsenin kymmenen säännön heuristiikkalistaa. Tällöin saatiin käytettyä kahta eri arviointitapaa käytettävyyden arviointiin. Näin pystyttiin testaamaan käytettävyys mahdollisimman kattavasti ja noudatettiin Nielsenin ohjetta kahden eri arviointitavan käytöstä. Alla on suoritetun testin havainnot kohta kohdalta.

- **Järjestelmän tilan näkyvyys**

  Käyttäjälle kerrotaan, mitä hänen tulisi tehdä missäkin vaiheessa. Käyttäjän antamalle syötteelle annetaan aina palaute onnistumisesta. Jos toiminto epäonnistuu, annetaan siitä virheilmoitus.

- **Järjestelmän ja todellisen maailman yhteys**

  Sovelluksessa käytetty käyttäjien äidinkieltä. Toiminnot ovat selkeässä järjestyksessä. Käyttöliittymän tekstit eivät sisällä erikoissanastoa, joten toiminnot ovat kaikkien ymmärrettävissä. Työntekijänumeron syöttämisessä on painikkeissa OK- ja Del-painikkeet. Nämä ovat kuitenkin niin vakiintuneita termejä, että kaikki ymmärtävät ne.

- **Käyttäjän hallinnan ja vapauden tunne**

  Ensiksi käyttäjän täytyy syöttää työntekijänumero. Tämän jälkeen käyttäjällä on vapaus valita mitä hän haluaa tehdä. Käyttäjällä on mahdollisuus päästä alkutilaan ”Vaihda työntekijä”-painikkeella jokaisesta muusta näkymästä, paitsi silloin kun luetaan viivakoodia. Tabletin alalaidassa olevasta järjestelmän paluu-painikkeesta käyttäjä

pääsee joka tilanteesta edelliselle näkymälle. Toiminnoista, joita ei voi enää perua, kysytään käyttäjältä varmistus. Tällaisia ovat työn poistaminen ja työn lopettaminen.

- **Johdonmukaisuus**

  Sovelluksen painikkeet sijaitsevat kokoajan samassa paikassa näytön ylälaidassa. Sovelluksessa eteneminen on johdonmukaista. Lisäksi tabletin alalaidan järjestelmän paluu-painikkeesta tapahtuu sama siirtyminen taaksepäin kuin muissakin Android-sovelluksissa.

- **Virheiden estäminen**

  Toiminnoista, joita ei voi perua pyydetään käyttäjältä varmistus. Jokainen toiminto on selkeä, eikä sekaantumisen vaaraa kahden toiminnon välillä ole. Kun käyttäjä on lukenut viivakoodin, näytetään hänelle luetun viivakoodin tuotteen ja työvaiheen kuvaukset, jolloin käyttäjä voi varmistua siitä, että on lukenut oikeat tiedot ennen työn aloittamista.

- **Tunnistettavuus ennen muistamista**

  Käyttöliittymän painikkeet ovat selkeitä, yksiselitteisiä tekstejä. Käyttäjän on helppo tietää, mitä mistäkin painikkeesta tapahtuu. ”Avoimet työt”-näkymässä työntekijä saa halutessaan näkyville tuotteet ja työvaiheen tiedot, joten hänen ei tarvitse muistaa tuotteiden ja työvaiheiden koodeja ulkoa.

- **Käytön joustavuus ja tehokkuus**

  Sovelluksen käyttö on helppoa niin aloittelijalle kuin edistyneelle käyttäjälle . Käyttö on luontevaa kosketusnäytön ansiosta. Kaikki tarvittavat toiminnot ovat helposti saatavilla jokaisessa näkymässä yläpalkin valikkorivin ansiosta. Työntekijännumero pysyy muistissa niin kauan, kunnes aletaan syöttää uutta työntekijänumeroa. Tämä nopeuttaa sovelluksen käyttöä silloin, kun vain yksi henkilö käyttää tablettia.

Toiminto ei kuitenkaan hidasta työntekijän vaihtoa, sillä siirryttäessä ”Vaihda työntekijä”-näkymään on työntekijänumero-kenttä aina valmiiksi tyhjänä.

- **Esteettinen ja minimalistinen suunnittelu**

  Koska sovellus perustuu näkymiin, on näkyvillä aina vain se, mitä käyttäjä tarvitsee kyseisen näkymän toiminnoissa. Tällöin käyttäjälle ei näytetä turhaa ja häiritsevää informaatiota.

- **Virheiden käsittely**

  Epäonnistuneesta toiminnosta kerrotaan käyttäjälle selkeästi virheen syy. Esimerkiksi, jos käyttäjä syöttää sellaisen työntekijänumeron, mitä ei ole olemassa, kerrotaan tästä hänelle heti. Jos syötetty työntekijänumero löytyy, ilmoitetaan ohjesivulla heti työntekijän nimi. Tällöin henkilö tietää onko syöttänyt oman työntekijänumeron. Jos käyttäjä on lukenut väärän viivakoodin, näkee hän heti ennen työnaloitusta tuotteen ja työvaiheen kuvaukset. Vaikka käyttäjä jatkaisi tästä eteenpäin on hänellä mahdollisuus vielä poistaa työ.

- **Opasteet ja dokumentaatio**

  Joka näkymässä on nähtävillä ohjeet ja opasteet siitä, mitä käyttäjän tulisi tehdä. Kaikilla muilla näkymillä ohjeet näkyvät suoraan ruudulla automaattisesti, paitsi ”Avoimet työt”-näkymässä, jossa on oikeassa ylälaidassa info-painike, jota painamalla avautuu ohje näkymän oikeaan laitaan.

6.4.2 Testauksen aikana syntyneet huomiot

Toisen version testauksen aikana saatu palaute oli tabletin suhteen positiivista, suurimmaksi eroksi sanottiin se, mitä osattiin odottaakin. Tabletin kanssa ei tarvinnut lähteä pois työpisteeltä, jolloin sen käyttö oli nopeampaa. Tablettia osattiin alun

opastuksen jälkeen myös käyttää hyvin, eikä sen käytön aikana tullut esille ongelmia käyttäjän kannalta.

Tabletin avulla kerättyjä tietoja analysoidessa huomattiin, että osa tehdyistä töistä kohdistui eriin, joita oltiin tehty jo aikaisemmin. Tällöin tabletilla kerättiin työaikoja vain osasta erää, mutta tehtyjä aikoja verrattiin koko erän kappalemäärään. Tämä tarkoitti sitä, etteivät saadut ajat per tuote olleet oikeita. Tämä ongelma ei olisi kuitenkaan aiheuttanut tulevaisuudessa enää ongelmia, koska uusien erien aloitus olisi tapahtunut silloin, kun tabletti olisi ollut käytössä.

7 LOPPUTULOS

Työn tarkoituksena oli löytää väline tai tapa kerätä tietoa tuotannosta niin, että sen avulla pystyttäisiin muodostamaan käsitys tietyn tuotteen tietyssä työvaiheessa kuluvasta ajasta. Kerätyn tiedon avulla oli tarkoitus saada laskettua tuotteen valmistukselle oikea hinta ja käyttää tietoa hyväksi tuotantoa suunnitellessa. Projekti alkoi käymällä lävitse erilaisia valmiita ratkaisuja, joita olisi voinut käyttää hyväksi. Alkukartoituksen jälkeen päätettiin kuitenkin lähteä rakentamaan järjestelmää itse.

Kehitetty järjestelmä saatiin toimivaksi, ja se täytti sille määritellyt vaatimukset. Vaatimuksina ollut helppokäyttöisyys tuli todettua käytettävyystestillä ja suorittamalla käytettävyyden arvio Nielsenin kymmenen säännön heuristiikkalistan avulla. Tuotantosolussa järjestetyn testin perusteella pystyttiin toteamaan, että halutut tiedot tallentuivat oikein ja niitä pystyttäisiin käyttämään. Järjestelmän raportointipäätteenä toimivat Android-tabletit maksavat noin 150€ kappale ja käytetyt palvelimet olivat käytössä jo ennestään. Käytössä olleet työkortit vaativat muutoksen, jolloin niihin liitettiin QR-koodit. Näin työkortteja pystyttiin käyttämään hyväksi, jolloin tuotteisiin ei tarvittu uusia tunnisteita. Edellä kerrotuista syistä järjestelmän kustannukset pysyivät pieninä, ja vaatimus kustannustehokkaasta ratkaisusta täyttyivät. Tabletin natiivi-sovelluksen ollessa hyvin minimaalinen ja näkymien lataaminen web-palvelimelta mahdollisti sen, että järjestelmää on helppo päivittää tarvittaessa. Järjestelmän luotettavuutta koitettiin parantaa käyttämällä HTML 5 local storage ratkaisua tuotteiden tietojen väliaikaiseen tallentamiseen. Käyttäjiltä saadun palautteen ja suoritetun heuristisen analyysin pohjalta voitiin todeta sovelluksen olevan helppokäyttöinen.
Järjestelmä olisi voitu ottaa halutessa käyttöön useammassakin tuotannonsolussa, mutta toistaiseksi näin ei ole kuitenkaan tehty.

LÄHTEET

[1] 2kmediat: Johdatus SQL:n maailmaan, verkkosivu. Saatavissa (viitattu 21.12.2014): www.2kmediat.com/sql/alkeet.asp

[2] AitoSolutions Oyj, verkkosivu. Saatavissa (viitattu 13.5.2015): http://www.aitosolutions.com/worktime/Default.cshtml

[3] Android UHF RFID Tablet, kuva, Saatavissa (viitattu 20.3.2015): http://www.cilico.us/wulu/uploads/allimg/130824/1-130R41540270-L.jpg

[4] Apache: About Apache Cordova, verkkosivu. Saatavissa (viitattu 16.1.2015): https://cordova.apache.org/

[5] Appgyver: Steroids, verkkosivu. Saatavissa (viitattu 16.1.2015): http://www.appgyver.com/steroids

[6] Avania: Käytettävyystestaus pähkinänkuoressa, verkkosivu. Saatavissa (viitattu 3.5.2015): http://www.avania.fi/kaytettavyystestaus-pahkinankuoressa/

[7] Barcode island: CODE 128 SYMBOLOGY, verkkosivu. Saatavissa (viitattu 2.5.2015): http://www.barcodeisland.com/code128.phtml

[8] Barcode Island: EAN-13 Background Information, verkkosivu. Saatavissa (viitattu 10.2.2015): http://www.barcodeisland.com/ean13.phtml

[9] Eshna – Simplilearn: The time confusion – Cycle Time, TAKT Time, Lead Time Part 1,verkkosivu. Saatavissa (viitattu 14.5.2015): http://www.simplilearn.com/time-confusion-cycle-time-takt-time-lead-time-part-1-article

[10] fastCGI, verkkosivu. Saatavissa (viitattu 21.12.2014): http://www.fastcgi.com/drupal/

[11] GitHub: FastClick, verkkosivu. Saatavissa (viitattu 15.5.2015): https://github.com/ftlabs/fastclick

[12] GS1 DataMatrix: An Introduction and Implementation Guideline. Saatavissa (viitattu 20.4.2015): http://www.gs1.org/docs/barcodes/GS1_DataMatrix_Introduction_and_technical_overview.pdf

[13] GS1 Finland: Viivakoodit. Saatavissa (viitattu 6.12.2014): http://www.gs1.fi/content/download/4705/30095/file/1.4+viivakooditaulu_suomi.pdf

[14] Heikki Hyyti: Automaatio- ja systeemitekniikan laboratoriotyöt, konenäkö robotin ohjauksessa. Saatavissa (viitattu 20.3.2015): https://noppa.aalto.fi/noppa/kurssi/as-0.2230/materiaali/AS-0_2230_tyon_3_tyoohje_ja_esiselostustehtavat.pdf

[15] History of PHP, verkkosivu. Saatavissa (viitattu 21.12.2014): http://php.net/manual/en/history.php.php

[16] ID Automation: Aztec Barcode FAQ and Tutorial, verkkosivu. Saatavissa (viitattu 1.5.2015): http://www.idautomation.com/barcode-faq/2d/aztec/

[17] ID Automation: Code 39 Barcode FAQ & Tutorial, verkkosivu. Saatavissa (viitattu 12.1.2015): http://www.idautomation.com/barcode-faq/code-39/

[18] Idesco Access Touch, kuva. Saatavissa (viitattu 20.3.2015): http://www.idesco.fi/wp-content/uploads/2013/06/AT-Card-Large.jpg

[19] Idesco: Access Touch. Saatavissa (viitattu 20.3.2015): http://www.idesco.fi/wpcontent/uploads/2013/05/Access-Touch-3.1-DatasheetEnglish.pdf

[20] ISO 9241-11: Guidance on usability. Saatavissa (viitattu 17.5.2015): http://www.iso.org/iso/catalogue_detail.htm?csnumber=16883

[21] Jakob Nielsen: 10 Usability Heuristics for User Interface Design, verkkosivu. Saatavissa (viitattu 2.5.2015): http://www.nngroup.com/articles/ten-usability-heuristics/

[22] Jakob Nielsen: Usability 101: Introduction to Usability, verkkosivu. Saatavissa (viitattu 2.5.2015): http://www.nngroup.com/articles/usability-101- introduction-to-usability/

[23] Jakob Nielsen: Characteristics of Usability Problems Found by Heuristic Evaluation, verkkosivu. Saatavissa (viitattu 17.5.2015): http://www.nngroup.com/articles/usability-problems-found-by-heuristic-evaluation/

[24] jQuery: What is jQuery?, verkkosivu. Saatavissa (viitattu 16.5.2015): http://jquery.com/

[25] Microsoft: The Official Microsoft ISS site, verkkosivu. Saatavissa (viitattu 21.12.2014): https://www.iis.net/

[26] Pervasive: database, verkkosivu. Saatavissa (viitattu 15.5.2015): http://www.pervasive.com/database

[27] PhoneGap, verkkosivu. Saatavissa (viitattu 16.1.2015): http://phonegap.com/

[28] RFID tag combo, kuva. Saatavissa (20.3.2015): http://files.hwkitchen.com/200000484-ca31ecb2b7/rfid-tag-combo.jpg

[29] RFIDLab Finland ry: RFID-tietoutta, verkkosivu. Saatavissa (viitattu 20.2.2015): http://www.rfidlab.fi/rfid-tietoutta

[30] SAP Crystal Reports, verkkosivu. Saatavissa (viitattu 20.2.2015): http://www.crystalreports.com/

[31] SFS verkkokauppa: SFS-EN ISO 9241-11, verkkosivu. Saatavissa (viitattu 2.5.2015): http://sales.sfs.fi/sfs/servlets/ProductServlet?action=productInfo&productID=144993

[32] TALtech: How a Barcode Reader Works, verkkosivu. Saatavissa (viitattu 25.4.2015): http://www.taltech.com/barcodesoftware/articles/how_barcode_reader_works

[33] TEC IT: Data Matrix (ECC200) - 2D Barcode, verkkosivu. Saatavissa (viitattu 20.4.2015): http://www.tecit.com/en/support/knowbase/symbologies/datamatrix/Default.aspx

[34] TEC-IT: QR-Code (2D Barcode), verkkosivu. Saatavissa (viitattu 20.1.2015): http://www.tec-it.com/en/support/knowbase/symbologies/qrcode/Default.aspx

[35] Teknologiateollisuus: Työtutkimuksen käsitteitä, menettelytapoja ja käyttökohteita. Saatavissa (viitattu 2.1.2015): https://teknologiateollisuus.fi/sites/default/files/file_attachments/tyomarkkinat_kannustava_palkkaus_palkkaustapoja_tyontutkimuksen_menettelytavat.pdf

[36] Trackway Manufacturing, verkkosivu. Saatavissa (viitattu 13.5.2015): http://www.trackway.eu/contpage.asp?sua=4&lang=1&s=480

[37] Unitech ms120 pen scanner, kuva. Saatavissa (viitattu 2.5.2015): http://c85c7a.medialib.glogster.com/media/42/4222154c9c18d2299053c99b13cd9afd2d3b541a25e4aa43935b20ea42d46315/22350414-260x260-0-0-unitech-ms120-pen-scanner-22350414-jpg.jpg

[38] Userfocus: ISO 9241 Part 11: Guidance on usability, verkkosivu. Saatavissa (viitattu 2.5.2015): http://www.userfocus.co.uk/resources/iso9241/part11.html

[39] Webopedia: ODBC-Open DataBase Connectivity, verkkosivu. Saatavissa (viitattu 20.12.2014): www.webopedia.com/TERM/O/ODBC.html

[40] Wikipedia, Konenäkö, verkkosivu. Saatavissa (viitattu 20.3.2015): http://fi.wikipedia.org/wiki/Konen%C3%A4k%C3%B6

[41] Wikipedia: Aztec Code, verkkosivu. Saatavissa (viitattu 21.3.2015): http://en.wikipedia.org/wiki/Aztec_Code

[42] Wikipedia: Barcode reader, verkkosivu. Saatavissa (viitattu 25.4.2015): http://en.wikipedia.org/wiki/Barcode_reader

[43] Wikipedia: Code 39, verkkosivu. Saatavissa (viitattu 12.1.2015): http://en.wikipedia.org/wiki/Code_39

[44] Wikipedia: Data Matrix, verkkosivu. Saatavissa (viitattu 20.4.2015): http://en.wikipedia.org/wiki/Data_Matrix

[45] Wikipedia: EAN-13, verkkosivu. Saatavissa (viitattu 20.4.2015): http://fi.wikipedia.org/wiki/EAN-13

[46] Wikipedia: Käytettävyys, verkkosivu. Saatavissa (viitattu 3.5.2015): http://fi.wikipedia.org/wiki/K%C3%A4ytett%C3%A4vyys

[47] Wikipedia: Käyttäjäkokemus, verkkosivu. Saatavissa (viitattu 3.5.2015): http://fi.wikipedia.org/wiki/K%C3%A4ytt%C3%A4j%C3%A4kokemus

[48] Wikipedia: Microsoft SQL Server, verkkosivu. Saatavissa (viitattu 21.12.2014): http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[49] Wikipedia: Pervasive PSQL, verkkosivu. Saatavissa (viitattu 15.5.2015): http://en.wikipedia.org/wiki/Pervasive_PSQL

[50] Wikipedia: QR Code structure, kuva. Saatavissa (viitattu 20.1.2015): http://fi.wikipedia.org/wiki/QRkoodi#/media/ File:QR_Code_Structure_Example_3.svg

[51] Wikipedia: QR Code, verkkosivu. Saatavissa (viitattu 20.1.2015): http://en.wikipedia.org/wiki/QR_code

LIITE A: ENSIMMÄISEN TESTIVERSION KYSELYLOMAKE

TUOTANNON SEURANTA

Järjestelmän 1. testaaminen lakkaamossa kesä-heinäkuu 2014

1. KÄYTTÖLIITTYMÄ JA LAITTEEN TOIMIVUUS

☐ Käyttöliittymä on hyvin selkeä ja helposti opittava.

☐ Käyttöliittymä on sekava, mutta helposti opittava.

☐ Käyttöliittymä on sekava ja vaikeasti opittava

☐ Laite on hidas (kerro kohdassa 3. mikä on hidasta)

☐ Tabletin näytön koko on tarpeeksi iso

☐ Laite tai ohjelma lopettaa toimintansa kesken kaiken

☐ Käyttöliittymää pystyy käyttämään sekä vasemmalla, että oikealla kädellä.

☐ Laitteen käyttö toimii paremmin sormella / kynällä (yli viivaa tarpeeton, älä yliviivaa kumpaakaan jos yhtä helppoa molemmilla).

☐ Viivakoodin luku on hidasta

☐ Viivakoodin luku on vaikeaa

2. KÄYTETTÄVYYS

☐ Tietojen syöttö on helppoa ja nopeaa

☐ Tietojen syöttö on helppoa, mutta hidasta

☐ Kosketusnäytön nappien painaminen on helppoa (napit tarpeeksi suuria ja riittävän erillään toisistaan. Kerro kohdassa 3 mitä pitäisi muuttaa jos tarvetta).

☐ Kosketusnäytön nappien painaminen on helppoa, mutta laite ei huomioi kaikkia painalluksia

☐ Nappien toiminnot ovat selkeät

☐ Tekstit ovat tarpeeksi isoja ja eri kenttien tekstit ovat selkeästi erillään

☐ Tekstit ovat tarpeeksi isoja, mutta voisivat olla eroteltuna toisistaan esim. viivoilla tai eri väreillä.

☐ Tekstin koko on liian pientä, mutta eri kenttien tekstit ovat kuitenkin selkeästi erillään.

☐ Ohjelma on niin selkeä, että sitä osaisi käyttää ilman erillistä koulutusta? (ohjelma opastaa tarpeeksi mitä pitäisi tehdä jne.. Kerro kohdassa 3 miten opastusta voisi parantaa.)

3. MITÄ TOIMINTOJA MUUTTAISIT JA MITEN?

esim. tekstien koko, värit, nappien koko, nappien välit, yms....

4. MITÄ TOIMINTOJA PITÄISI LISÄTÄ TAI POISTAA?

Ideoita järjestelmän kehittämiseen.

5. MIELIPITEITÄ JÄRJESTELMÄN TARPEELLISUUDESTA

Onko järjestelmä tarpeellinen vai meneekö se päällekkäin jo käytössä olevan kanssa, voisiko tapoja yhdistää tai
mikä pitäisi korvata ja millä?