KEMI-TORNION AMMATTIKORKEAKOULU

Työpöytäsovellus yrityksen tarvikekannan hallintaan

Jenni Lounela

Tietojenkäsittelyn koulutusohjelma
Web-asiantuntija
Tradenomi

TORNIO 2010

TIIVISTELMÄ

Lounela, Jenni 2010. Työpöytäsovellus yrityksen tarvikekannan hallintaan. Opinnäytetyö. Kemi-Tornion ammattikorkeakoulu. Kaupan ja kulttuurin toimiala. Sivuja 77. Liitteet 1-17.

Opinnäytetyössäni tutkin työpöytäsovelluksen kehittämistä. Toimeksiantajayritys, kemiläinen Arkkitalo Rintala, on huomannut käytännön ongelman laajan tarvikekannan hallintaan liittyen. Ongelma pyritään ratkaisemaan tietokantapohjaisella työpöytäsovelluksella. Sovelluksella pyritään vastaamaan myös toimeksiantajayrityksen työtehtävien suunnittelu- ja valmisteluongelmaan.

Työssä käytän konstruktiivista tutkimusmenetelmää ja tutkin sovelluskehitystä pääosin käytettävyyden näkökulmasta. Käytettävyyteen liittyviä huomioita poimin muun muassa Suomen lainsäädännöstä ja kansainvälisistä ISO-standardeista. Työpöytäsovelluksen prototyypin ohjelmoin Visual Studio-ohjelmointiympäristössä ja sen pohjalle tuleva tietokanta rakennetaan Microsoftin Access-tietokantaohjelmalla. Sovelluksen kehitystyö etenee opinnäytetyössäni vesiputousmallin mukaisesti.

Työn tuloksena on toimiva prototyyppi työpöytäsovelluksesta. Prototyyppiä käytetään sovelluksen kehityksen pohjana. Olen tiedostanut prototyyppiin ja sen kehitykseen liittyvät puutteet ja ongelmat, ja pohdin opinnäytetyössäni niihin liittyviä kehitys- ja ratkaisutapoja.

Prototyyppi vastaa sille asetettuihin päävaatimuksiin. Sitä ei kuitenkaan ole tarkoitettu käytettäväksi ennen kuin siihen kohdistetaan opinnäytetyössäni mainittuja jatkokehitystoimenpiteitä.

Asiasanat: Visual Basic, relaatiotietokanta, käyttöliittymä, työpöytäsovellus, käytettävyys, prototyyppi

ABSTRACT

Lounela, Jenni 2010. Desktop application for business-associated device management. Bachelor's thesis. Kemi-Tornio University of Applied Sciences. Pages 77. Appendices 1-17.

In my thesis I do research on how to develop a desktop application. My assignor, Arkkitalo Rintala from Kemi, has noticed a problem in practice regarding managing the vast amount of photographic devices owned by the company. The goal is to solve the problem with a database-driven desktop application. The application should also solve the problem with planning and preparing work-related tasks.

Using constructive research method I research application development mainly from the viewpoint of usability. I search perspectives for usability for example from the legislation of Finland and the international ISO standards. I build the desktop application prototype in Visual Studio programming environment and the database in Microsoft Access database application. The development of the application proceeds according to the Waterfall Model.

The result of this thesis is a functioning desktop application prototype. I acknowledge the deficiencies and problems in the prototype and its development. I look for appropriate development methods and solutions for the prototype in my thesis.

The prototype meets the defined main requirements. However, it is not to be used until it is further developed according to at least the instructions given in this thesis.

Keywords: Visual Basic, relational database, graphical user interface, desktop application, usability, prototype

SISÄLTÖ

1 JOHDANTO

Yrityksen pyörittämisessä ja liiketoimintaprosessien suorittamisessa sekä yritys että sen työntekijät käyttävät erilaisia tarvikkeita. Näitä tarvikkeita voivat olla yrityksen toimialasta, suoritetuista tehtävistä ja tarjotuista palveluista riippuen tietokoneet, puhelimet, kopiokoneet tai erilaiset korjaukseen ja rakentamiseen liittyvät tarvikkeet. Mitä suurempi yritys tai tarvikekanta on kyseessä, sitä tärkeämmäksi muodostuu tarvikekannan tehokas hallinta. Kun paperiarkisto tai Excel-taulukko ei enää riitä tarvikekannan hallintaan, on syytä alkaa pohtia muita vaihtoehtoja.

Tarvikkeiden eli resurssien hallintaan on olemassa erilaisia valmiita sovelluksia. Myös edellä mainitulla Excel-taulukolla voidaan hallita hyvinkin kattavia tarvike- ja muita resurssikantoja. Vaikka valmiita vaihtoehtoja on monia, voivat hallintasovelluksen tarvitsijan vaatimukset aiheuttaa tarpeen luoda täysin yksilöllinen ratkaisu.

Opinnäytetyöni aiheena on yrityksen tarvikekannan hallintaan tarkoitetun työpöytäsovelluksen kehittäminen. Ratkaisua tarvikekannan hallintaan pyysi kemiläinen yrittäjä Seppo Rintala. Rintalan yritys Arkkitalo Rintala on pääasiallisesti arkkitehtipalveluja tarjoava yritys. Osa liiketoiminnasta muodostuu myös valokuvaamisesta ja kuvien myynnistä. Valokuvausta Rintala hyödyntää myös arkkitehdin työssään. Toimeksiantaja haluaakin mahdollisimman yksinkertaisen ja toimivan ratkaisun valokuvaustarvikekannan hallintaan. Toiveena on myös yksinkertaista linjaa noudattava ratkaisu kuvaustehtävien suunnitteluun ja valmisteluun.

Konstruktiivisella tutkimusmenetelmällä, jota työssäni käytän, tuotetaan ratkaisuja määriteltyihin ongelmiin (Konstruktiivinen tutkimusote 2006). Useimmiten ongelma tai toimimaton ratkaisu havaitaan käytännössä (Seppänen 2004). Tässä tapauksessa toimeksiantajan ongelmana on laajan tarvikekannan riittämätön ja vaikea hallinta sekä työtehtävien suunnittelu ja valmistelu. Ongelma on selkeästi käytännössä havaittu. Tarvikekantaa on ylläpidetty Excel-taulukossa, jonka tarjoamat mahdollisuudet on huomattu rajallisiksi tarvikekannan kasvaessa. Konstruktiivista tutkimusmenetelmää hyväksikäyttäen tavoitteena on tuottaa ongelmaan ratkaisu, joka helpottaa yrittäjän arkea ja työntekoa. Parhaimmillaan ratkaisu vapauttaa aikaa tarvikekannan hallinnalta ja työtehtävien suunnittelulta muille tehtäville.

Toimeksiantajan vaatimuksiin ja toiveisiin parhaiten vastaa työpöytäsovellus, jonka pohjalle tulee tietokanta. Sovelluksen käyttöliittymästä tehdään mahdollisimman yksinkertainen ja helppokäyttöinen, kuitenkin pitäen mielessä vaatimuksen suuren tietomäärän käsittelyyn. Sovelluksen avulla toimeksiantaja pystyy lisäämään, muokkaamaan ja poistamaan tarviketietoja. Sovelluksen yksi ominaisuus on myös tallennettava ja tulostettava ”ostoskori”, joka mukailee verkkokaupoistakin tuttua mallia. Kori helpottaa kuvaustehtävien suunnittelua ja valmistelua. Lisättäessä tarvikkeita koriin ohjelma laskee tarvikkeiden kokonaispainon ja huomauttaa tarvittaessa tarvikkeiden yhteensopivuudesta. Käyttöympäristönä tulee olemaan ainakin Windows XP Home- ja Windows 7- käyttöjärjestelmät.

Sovelluksen tulen toteuttamaan Visual Basic -ohjelmointikielellä Visual Studio 2008 -ohjelmointiympäristössä. Molemmat ovat Microsoftin kehittämiä ja niiden avulla voidaan luoda erilaisia sovelluksia pääosin Microsoft Windows -ympäristöön (FunctionX 2010). Visual Studio on nykyaikainen ja monipuolinen sovelluskehitin. Se sisältää muun muassa koodieditorin, visuaaliset suunnitteluvälineet ja virheenjäljittäjän. Visual Studiolla voidaan tuottaa nopeasti valmiita sovelluksia käyttäen erilaisia ohjelmaan sisällytettyjä toimintoja, kuten tietokantoja. (Järvinen 2008, 2-3.)

Opinnäytetyössäni tulen tutkimaan mikä on paras tietokantaohjelmisto työpöytäsovelluksen pohjalle. Olen valinnut kaksi ohjelmistovaihtoehtoa, joista toinen on avoimen lähdekoodin ilmainen OpenOffice Base -ohjelma ja toinen Microsoftin maksullinen Access-ohjelma.

Käyttöliittymän suunnittelussa hyödynnän GUIDe-prosessimallia. GUIDe-prosessimalli tarkoittaa käyttöliittymän suunnitteluprosessia, jossa käyttöliittymäsuunnittelu tapahtuu projektin alussa. GUIDe-prosessimallin etuna on käyttöliittymäratkaisujen testaus varhaisessa vaiheessa, jolloin tarvittavien muutosten toteuttaminen on helppoa.

Suunnittelun työkaluna käytän muun muassa graafista UML-mallinnuskieltä. UML-mallinnuskielen erilaisilla osilla voidaan määritellä ja dokumentoida oliopohjaisia ohjelmointikieliä, kuten Visual Basicia (Kankaanpää 2010).

Tutkimuskysymykseni käsittelevät sovelluksen käytettävyyttä ja käytön ohjausta. Käytettävyyttä tarkastelen muun muassa lain ja kansainvälisten ISO-standardien näkökul-

masta, unohtamatta kuitenkaan tunnetun käytettävyysasiantuntijan Jakob Nielsenin oppeja.

Opinnäytetyöni alussa käsittelen tutkimusasetelmaa muun muassa valitsemani tutkimus- ja tiedonkeruumenetelmän valossa. Tämän jälkeen tarkastelen ohjelmiston kehitystyön vaiheita ja periaatteita. Suuri osa opinnäytetyötäni on tietokannan ja käyttöliittymän suunnittelu ja toteutus, joten näitä asioita tulen myös käsittelemään. Koska opinnäytetyönä luon tehtävän sovelluksen prototyypin, käsittelen myös sovelluksen jatkokehityksen kannalta huomioonotettavia asioita. Opinnäytetyöni lopetan työn tulosten analysoinnilla.

2 TUTKIMUSASETELMA

Tässä luvussa käsittelen opinnäytetyöni tutkimusasetelmaa. Tutkimusmenetelmänä käytän konstruktiivista tutkimusmenetelmää, jota käytetään paljon tietojenkäsittelyn alan tutkimuksissa. Konstruktiivinen tutkimusmenetelmä on paikallaan uusia sovelluksia ja ratkaisuja luotaessa. Opinnäytetyön tuloksena tulee olemaan toimeksiantajan vaatimusten, toiveiden ja tarpeiden pohjalta rakennettu työpöytäsovelluksen prototyyppi yrityksen tarvikekannan hallintaan. Samankaltaisia sovelluksia ja ohjelmistoja on jo olemassa, mutta tekemäni sovellus räätälöidään nimenomaan yhden yksittäisen yrityksen ja käyttäjän tarpeita silmällä pitäen. Työssä tarkemmin esittelemäni OpenOffice Base ja Microsoft Access tarjoavat samankaltaisia toimintoja ja ominaisuuksia, kuin tekemäni työpöytäsovellus. Näiden ohjelmien käyttäminen sellaisenaan, ilman erillistä sovellusta, vaatii käyttäjältä tietämystä sekä tietokantojen suunnittelu- ja toteutustavoista että tietokantaohjelmiston hallinnasta ja lukuisista ominaisuuksista. Toimeksiantajallani ei ole aikaa eikä mielenkiintoa alkaa paneutua tämänkaltaisiin asioihin.

Tehtävän työpöytäsovelluksen tarkat ominaisuudet ja toiminnot sisältävän vaatimusmäärittelyn olen rakentanut palavereissa suorittamieni avoimien haastattelujen pohjalta. Yksi keskeisistä vaatimuksista onkin sovelluksen yksinkertaisuus niin toimintojen kuin käyttöliittymän suhteen. Tähän vaatimukseen eivät edellä mainitut tietokantaohjelmat pysty vastaamaan. Tämän vuoksi täysin uusi ja ennalta määrättyyn tarkoitukseen yksilöity ja rajattu sovellus on paikallaan.

Opinnäytetyöni tutkimuskysymykset olen pyrkinyt asettamaan niin, että ne tukevat ja ohjaavat työskentelyä ja sovelluksen kehittämistä. Tutkimuskysymykset koskevat sekä tutkimuksen että työn kannalta keskeisiä asioita eli tietokantaa, käyttöliittymää, käytön opastusta ja sovelluksen käytettävyyttä.

2.1 Tutkimusmenetelmä

Tutkimusmenetelmänä käytän konstruktiivista tutkimusmenetelmää. Konstruktiiviselle tutkimusmenetelmälle ominaista on johonkin sovellutukseen tai tavoitteeseen tähtäävän uuden tiedon tuottaminen. Tutkimuksella tuotetaan ratkaisuja tai konstruktioita määriteltyihin ongelmiin. Konstruktiivisen tutkimusmenetelmän ominaisuuksia ovat aiemman

tietämyksen hyödyntäminen ongelmanratkaisussa sekä tuotetun ratkaisun toimivuuden ja uutuuden osoittaminen. (Konstruktiivinen tutkimusote 2006.) Uudeksi tiedoksi voidaan kutsua myös tutkijan tuottamaa tietoa, jossa pystytään osoittamaan aiemman tiedon käyttömahdollisuuksia uuden toiminnan kehittämisessä tai aiemman tiedon yhdistelemistä uusilla tavoilla. Varsinkin työelämässä hyödytään tutkimuksista, joissa kuvataan olemassa olevan tiedon uudenlaista käyttämistä tai yhdistelyä. (Vilkka 2005, 23.)

Konstruktiivista tutkimusmenetelmää käytetään esimerkiksi ohjelmistojen tai laitteiden suunnitteluun ja arviointiin. Tutkimusmenetelmään liittyvä arviointi tai evaluointi voidaan kohdistaa lopputulokseen ja sen ominaisuuksiin, konstruktioprosessiin tai käytettyyn tekniikkaan. (Seppänen 2004.)

Usein konstruktiiviselle tutkimukselle ja sen kautta toteutetulle ratkaisumallille tulee tarve käytännössä havaituista puutteista, ongelmista tai toimimattomista ratkaisuista. Tutkimuksessa korostuukin tilaajan tai hyödyntäjän ja toteuttajan välinen kommunikaatio. Usein käytännön edustajille on haasteellista sisällyttää teoreettista tietämystä ongelmanratkaisuun. Teorian ja käytännön yhdistäminen onkin tärkeä osa konstruktiivista tutkimusmenetelmää käyttävän tutkijan työtä. (Seppänen 2004.)

Konstruktiivinen tutkimus etenee vaiheittain. Tutkimus aloitetaan etsimällä tutkimuksellisesti mielenkiintoinen ongelma ja hankkimalla tutkittavasta kohteesta esiymmärrys esimerkiksi lähdemateriaaleihin tutustumalla. Tämän jälkeen voidaan ratkaisumallia alkaa suunnitella ja konstruoida. Tutkimus päätetään ratkaisun toimivuuden testauksella sekä käytettyjen teoriakytkentöjen ja lopputuloksen tieteellisen uutuusarvon osoittamisella. (Konstruktiivinen tutkimusote 2006.)

Konstruktiivisen tutkimusmenetelmän käytöstä hyötyy sekä tilaaja että tutkija. Tutkimuksen hyötyjä ovat esimerkiksi käytännön ongelman tieteellinen analysointi ja ratkaisu, kehitystyön ja tutkimuksen välisen kuilun pienentäminen, molempien osapuolten sitoutuminen hankkeeseen ja tutkimuksen ratkaisun kattava testaus. Konstruktiiviseen tutkimukseen sisältyy myös mahdollisia riskejä, kuten esimerkiksi tutkimuksen pitkäkestoisuus, teoriaperustan puutteellisuus ja erilaisten resurssien, kuten tietotaidon, rahoituksen ja tarvittavan henkilöstön puute. (Seppänen 2004.)

Oli käytetty tutkimusmenetelmä mikä tahansa, on tutkimuksen tekemisessä noudatettava tutkimusetiikkaa ja hyvää tieteellistä käytäntöä. Tutkimusetiikka pitää sisällään yleisesti sovittuja sääntöjä, koskien esimerkiksi tutkimuskohdetta ja toimeksiantajia. Kun tutkimuksen tekemisessä käytetään tiedeyhteisön hyväksymiä tiedonhankinta- ja tutkimusmenetelmiä, voidaan puhua hyvän tieteellisen käytännön noudattamisesta. Tiedonhankinnan suhteen hyvä tieteellinen käytäntö tarkoittaa tiedonhankinnan perustamista tieteellisen kirjallisuuden tuntemukselle sekä muihin tietolähteisiin ja oman tutkimuksen analysointiin. Hyvän tieteellisen käytännön mukaisesti tutkimuksen on tuotettava uutta tietoa tai osoitettava olemassa olevan tiedon uusia hyödyntämis- tai yhdistelytapoja. Tutkijan tulee noudattaa rehellisyyttä, huolellisuutta ja tarkkuutta tutkimuksen kaikissa vaiheissa. (Vilkka 2005, 30.)

2.2 Tiedonkeruumenetelmä

Työssä tarvittavia tietoja saan toimeksiantajalta sekä keräämistäni kirjallisuus- ja muista lähteistä. Koska toimeksiantajayrityksessä on vain yksi työntekijä ja tehtävä sovellus tulee vain hänen käyttöönsä, on paras tietojen keräysmenetelmä haastattelu.

Haastattelun etu, muihin tiedonkeräysmenetelmiin verrattuna, on mahdollisuus säädellä joustavasti aineiston keruuta tilanteen ja vastaajan mukaisesti. Haastattelussa on helppo muokata aiheiden järjestystä ja tulkita annettuja vastauksia. Haastattelutilanteessa vastaaja on aktiivinen osapuoli ja hän voi kertoa aiheeseen liittyviä mielipiteitään vapaammin kuin esimerkiksi kyselylomaketta käytettäessä. Saatavia vastauksia on helppo selventää haastattelussa ja tarvittaessa haastatteluaiheita voidaan näin laajentaa ja monipuolistaa riittävän lopputuloksen saavuttamiseksi. (Hirsjärvi & Remes & Sajavaara 2009, 205.)

Monien etujen lisäksi haastattelun käytössä on myös ongelmia ja haasteita. Haastattelu on aikaa vievää ja tilannesidonnaista. Haastattelun käyttö vaatii suunnittelua ja haastattelijan roolin omaksumista ja siihen kouluttautumista. Haastateltava voi vastata kysymyksiin tavalla, jonka ajattelee olevan sosiaalisesti hyväksyttävä. Haastattelijan tuleekin kyetä tulkitsemaan haastateltavan vastauksia paremmin kuin esimerkiksi kyselylomaketta käytettäessä. (Hirsjärvi ym. 2009, 206-207.)

Tutkimushaastattelun tyypit voidaan jakaa kolmeen pääryhmään: strukturoituihin haastatteluihin, teemahaastatteluihin ja avoimiin haastatteluihin. Strukturoitu haastattelu tapahtuu kyselylomakkeen avulla. Teemahaastattelussa kysymysten tarkka muoto ja järjestys puuttuvat, mutta haastattelun aihepiirit tai teema-alueet ovat tiedossa. Avoimessa haastattelussa mielipiteitä, käsityksiä ja ajatuksia selvitetään niiden tullessa aidosti vastaan keskustelussa. Avoin haastattelu, joka on haastattelutyypeistä lähimpänä keskustelua, on mielestäni paras tiedonkeruumenetelmä tätä työtä ajatellen. (Hirsjärvi ym. 2009, 208-209.)

Avoin haastattelu vaatii tavallisesti paljon aikaa ja useita haastattelukertoja. Avoin haastattelu kestääkin yleensä pari tuntia. Avoimessa haastattelutilanteessa tilanteen ohjaus on haastattelijan vastuulla. Haastattelija selvittää haastateltavan mielipiteitä, käsityksiä ja ajatuksia sen mukaisesti, miten ne tulevat keskustelussa vastaan. Haastattelun toteutuksessa on otettava huomioon muun muassa haastateltavan puheliaisuus tai niukkasanaisuus, keskustelun avaukset, kysymykset ja keskustelun ohjaaminen. (Hirsjärvi ym. 2009, 209, 211.) Avoimen haastattelun lisäksi suorittamaani tiedonkeruumenetelmää voidaan kutsua yksilöhaastatteluksi, koska haastateltavana on vain yksi henkilö eli toimeksiantajayrittäjä Seppo Rintala. Tiedonkeruuta avointa haastattelua käyttäen olen suorittanut toimeksiantajan kanssa pidetyissä palavereissa. Palavereissa on käyty läpi asioita välillä tietyn asiakokonaisuuden mukaan, välillä taas sen mukaisesti kun asiat ovat tulleet keskustelussa esiin. Ennalta määrittelemiäni asiakokonaisuuksia ovat muun muassa sovelluksen tarkoitus, sovelluksen toiminnot, käyttöliittymän rakenne ja tarviketietojen jaottelu. Toisinaan tiedonkeruun ja avoimen haastattelun tukena olen käyttänyt esimerkiksi toimeksiantajan Excel-tarviketaulukkoa, käyttöliittymäsuunnitelmaani sekä tekemääni prototyyppiä.

2.3 Tutkimuskysymykset

Perinteisesti tutkimuksen tekemisessä on korostettu tutkimusongelmien tai -kysymyksien harkitsemista ja selkeää muotoilua ennen aineiston keräämistä. Usein ongelmien asettaminen ja niiden muotoileminen on kuitenkin vaikeampaa kuin niiden ratkaiseminen. Tutkimuksessa voidaankin varautua tutkimusongelman muuttumiseen tutkimusprosessin aikana. Tästä huolimatta tutkimusongelmasta tai -kysymyksistä on olta-

va jonkinlainen käsitys, koska ilman spesifioitua kysymyksenasettelua tutkimus jää aineiston luokittelun tasolle. (Hirsjärvi ym. 2009, 125-126.)

Tutkimukseen pitää pystyä löytämään juoni tai johtoajatus. Johtoajatuksessa yhdistyy tutkimuksen perusidea, tarkoitus, näkökulma ja käsittelyn rajaus. Johtoajatuksen mukaisesti pystytään muodostamaan tutkimuksen pääongelma. Yleensä pääongelma on yleisluontoinen kysymys, joka hahmottaa tutkittavan kokonaisuuden. Tutkimuksen osa- tai alaongelmat saadaan täsmentämällä pääongelmaa. Vastaukset alaongelmiin mahdollistavat myös pääongelmaan vastaamisen. Tutkimusongelmat esitetään yleensä kysymyksen muodossa. (Hirsjärvi ym. 2009, 41, 126-129.)

Olen asettanut opinnäytetyötä varten kolme tutkimusongelmaa tai -kysymystä, joista yhteen liittyy tarkentavia kysymyksiä. Tutkimuskysymykseni ovat:

1. Mikä on työpöytäsovellukseen sopiva tietokantaohjelma?
2. Miten käyttöliittymässä yhdistetään helppokäyttöisyys ja suuren tietomäärän käsittely?
3. Millainen käyttöohje palvelee sovelluksen käyttäjää parhaiten?
   - Miten kirjallisesta käyttöohjeesta saadaan mahdollisimman yksinkertainen ja selkeä?
   - Miten itse sovellukseen lisätään yksinkertainen ja selkeä käytön ohjeistus?

Asettamissani tutkimuskysymyksissä yhdistyy sekä lopputuotteen kannalta tärkeät seikat että itseäni eniten kiinnostavia asioita tietojenkäsittelyn alalta. Ensimmäiseen tutkimuskysymykseen pyrin vastaamaan tutkittuani erilaista lähdemateriaalia ja tutustuttuani erilaisiin vaihtoehtoihin käytännössä. Toisen tutkimuskysymyksen ratkaisua pyrin helpottamaan GUIDe-prosessimallia apuna käyttäen. Kolmanteen tutkimuskysymykseen haen vastausta muun muassa käytettävyysasiantuntija Jakob Nielsenin teoksista.

3 NÄKÖKULMANA KÄYTETTÄVYYS

Tieteellisessä tutkimuksessa teoriaa käytetään uuden tiedon tuottamiseen. Tutkittavaa asiaa tarkastellaan teorian kautta. Voidaankin puhua teoreettisesta viitekehyksestä tai teoreettisesta lähestymistavasta. Aiemmissa tutkimuksissa ilmenneet säännönmukaisuudet määrittävät, selittävät ja lisäävät tutkittavaa asiaa koskevaa ymmärrystä. Erilaiset teoriat ja käsitteet eivät ole käytännöstä irrallisia, vaan ne ovat muodostuneet käytännön tutkimustoiminnasta. Teoriaa sovelletaan käytäntöön ja käytäntö muodostaa teorioita. Teorialla voidaan ohjata ja korjata kyseenalaisia käytäntöjä. (Vilkka 2005, 24-25.)

Opinnäytetyössäni otan teoreettiseksi näkökulmaksi ohjelmistojen käytettävyyden. Ohjelmiston käytettävyys määräytyy pitkälti tehdyn käyttöliittymän mukaan. Käyttöliittymän avulla päästään käyttämään ohjelmiston sisältämää tietoa tai palvelua. Käyttöliittymä sisältääkin esimerkiksi tavat navigoida eri sivuilla ja tiedon etsintä. Käytettävyyttä tai helppokäyttöisyyttä tarkastellaan sekä uuden käyttäjän että toistuvasti ohjelmistoa käyttävän henkilön näkökulmasta. Käytettävyyden mittaamisessa huomiota kiinnitetään muun muassa ohjelmiston käytön opetteluun sekä toistuvien toimenpiteiden nopeuteen. (Korpela & Linjama 2003, 360.)

Tiettyjä suosituksia kannattaa yrittää noudattaa käyttöliittymää ja sen toimintoja suunniteltaessa ja toteutettaessa. Näitä suosituksia ja ohjeita antavat muun muassa laki, kansainvälisesti määritellyt standardit sekä useat käytettävyyden asiantuntijat, joista tunnetuimpia lienee Jakob Nielsen.

Käytettävyyttä ja ohjelmistojen ominaisuuksia säätelee Suomessa laki. Valtioneuvoston vuonna 1993 tekemässä päätöksessä annetaan määräykset näyttöpäätetyöstä. Päätöksen vähimmäisvaatimuksia käsittelevässä liitteessä annetaan ohjeistukset, jotka koskevat tietokoneen käyttäjäliittymää. Nämä lainmääräykset tulee ottaa huomioon ohjelmistojen suunnittelu- ja toteutusvaiheissa, vaikka valtioneuvoston päätöksessä vastuulliseksi tahoksi mainitaankin työnantaja. Pääpiirteittäin laki määrää, että ohjelmiston ja sen käyttöliittymän on sovittava suoritettavaan tehtävään, oltava helppokäyttöinen, annettava palautetta käyttäjän toimista, näytettävä käsiteltävä tieto käyttäjän ymmärtämällä tavalla ja noudatettava ohjelmistoergonomian periaatteita. (Valtioneuvoston päätös näyttöpäätetyöstä 1983.)

Valtioneuvoston päätöksessä kirjattuja ohjelmistojen käytettävyyden vaatimuksia syvällisemmin kuvaavat esimerkiksi kansainväliset ISO-standardit ja käytettävyysasiantuntija Jakob Nielsenin opit. Näitä molempia käsittelen tarkemmin valottaakseni erilaisia näkökulmia ohjelmistojen käytettävyyteen ja helppokäyttöisyyteen liittyen.

3.1 ISO-standardit

ISO eli International Organization for Standardization on maailman suurin kansainvälisten standardien kehittäjä ja julkaisija. Järjestö on perustettu vuonna 1947. ISOn jäseniä ovat kansalliset standardoimisjärjestöt, joita on yhteensä 163 maasta. Suomen edustaja ISOssa on Suomen Standardisoimisliitto SFS. Järjestön keskus sijaitsee Sveitsissä. ISO toimii yleisen ja yksityisen sektorin välissä, ja varmistaa näin ratkaisujen sopivuuden molemmilla sektoreilla. Perustamisen jälkeen ISO-järjestö on kehittänyt yli 18000 kansainvälistä standardia. Standardien kohteina on suuri määrä eri aloja, kuten maanviljely, rakentaminen, lääketiede ja it-ala. (ISO 2010.)

ISO-järjestön kehittämistä standardeista tunnetuimpia ovat ISO 9000-sarjan standardit. 9000-sarjan standardeja voidaan käyttää kaiken kokoisissa ja tyyppisissä organisaatioissa. ISO-standardien käyttö mahdollistaa systemaattisen tavan ohjata organisaation eri prosesseja. Standardit ovat suosituksia ja niitä voidaan käyttää sekä tavara- että palvelutuotteen tai näiden yhdistelmän laadun kuvaamisessa ja varmistamisessa. Standardien avulla organisaatio kykenee menestyksekkäästi täyttämään asiakkaiden sekä sidosryhmien odotukset ja tarpeet. (Frost 2007.)

ISO 9000-sarja sisältää standardeja tietotyön ergonomian yleisperiaatteista ja ohjelmistoista. ISO 9241-sarjan standardit määrittävät näyttöpäätteillä tehtävän toimistotyön ergonomiset vaatimukset. Standardisarjan osa 11 keskittyy käytettävyyden määrittelyyn ja arviointiin, kun taas osa 10 sisältää dialogin periaatteet. (Suomen standardisoimisliitto 2000, 61, 151.)

ISO 9241–11-standardi keskittyy käytettävyyden määrittelyyn ja arviointiin järjestelmän käyttäjän suoriutumisen ja tyytyväisyyden näkökulmasta. Standardia voidaan hyödyntää näyttöpäätteiden ja tietojärjestelmien käytettävyyden suunnittelussa ja arvioinnissa. Mittausta suoritetaan haluttujen tavoitteiden saavuttamisella, tavoitteiden saavut-

tamiseen vaaditulla työmäärällä ja käyttäjän kokemalla käyttömukavuudella. (Suomen standardisoimisliitto 2000, 66.)

ISO 9241–11-standardi määrittelee käytettävyyden seuraavasti: ”Mitta, miten hyvin määrätyt käyttäjät voivat käyttää tuotetta määrätyssä käyttötilanteessa saavuttaakseen määritetyt tavoitteet tuloksellisesti, tehokkaasti ja miellyttävästi”. Standardin liitteessä huomautetaan käytettävyysominaisuuksien riippuvan tuotteen käyttäjän, tehtävän ja ympäristön luonteesta. Käyttöominaisuudet ja käytettävyys liittyvät määrättyyn käyttötilanteeseen, eikä tuotteella näin voida katsoa olevan luontaista käytettävyyttä. (Suomen standardisoimisliitto 2000, 66, 96.)

Tuotteen käytettävyyttä voidaan mitata esimerkiksi tuloksellisuuden ja tehokkuuden analysoinnilla, käytettäessä tuotetta määrätyssä käyttötilanteessa. Tähän liittyy myös tuotteen käyttäjän tyytyväisyyden mittaus. Standardissa paljon käytetty sana tyytyväisyys on määritetty tarkoittavan epämukavuuden puuttumista ja myönteistä suhtautumista tuotteen käyttöön. (Suomen standardisoimisliitto 2000, 66, 96.)

ISO 9241–11-standardin mukaan tuotteen, eli tässä tapauksessa sovelluksen, käytettävyyttä voidaan parantaa lisäämällä tuotteeseen ominaisuuksia, jotka hyödyttävät käyttäjiä tietyissä käyttötilanteissa. Käytettävyyttä suunniteltaessa on otetta huomioon myös käyttäjien merkitykselliset piirteet, kuten tieto- ja taitotaso, koulutus, harjaantuneisuus ja motoriset ja aisteihin liittyvät kyvyt. (Suomen standardisoimisliitto 2000, 68, 70.)

ISO 9241-standardisarjan osa 10 määrittelee yleisellä tasolla päätetyöhön tarkoitettujen ohjelmistojen ergonomista suunnittelua ja ergonomisia periaatteita. Standardin soveltaminen mahdollistaa käyttökelpoisempien, johdonmukaisempien ja tuottavuutta parantavien käyttöliittymien suunnittelemisen ja toteuttamisen. (Suomen standardisoimisliitto 2000, 154.)

ISO 9241–10-standardin mukaan käyttöliittymäsuunnittelun tärkeimpiä periaatteita ovat muun muassa sopivuus työtehtävään, hallittavuus ja yhdenmukaisuus käyttäjän odotuksiin nähden. On myös huomioitava, että järjestelmän on annettava käyttäjän suorittamiin toimenpiteisiin tarkoituksenmukaista palautetta. (Suomen standardisoimisliitto 2000, 154, 158.)

3.2 Jakob Nielsenin käytettävyysmääritelmiä

Jakob Nielsen on kansainvälisesti tunnettu käytettävyysasiantuntija. Hän painottaa käytettävyysopeissaan web-sivustojen käytettävyyttä ja käytettävyyden sekä käyttöliittymän suunnittelua. Nielsenin käytettävyysoppeja voidaan kuitenkin soveltaa myös ohjelmistokehitykseen, koska samat käytettävyyden periaatteet toistuvat niin internetissä kuin sovelluksissakin.

Jakob Nielsenin mukaan käyttöliittymäsuunnittelussa huomio tulee kiinnittää käyttöliittymän eri elementteihin. Kun käyttöliittymäsuunnitelmaa tarkastellaan poistamalla siitä kaikki elementit yksi kerrallaan, huomataan käyttöliittymässä mahdollisesti olevat ylimääräiset ja toiminnan kannalta turhat elementit. (Nielsen 2000, 22.) On parempi käyttää vain muutamaa toimintoja avustavaa elementtiä kuin suurta määrää toimintoja hidastavia elementtejä (Nielsen & Loranger 2006, 384). Käyttöliittymissä yksinkertaisuus ja yksinkertaistaminen ovat aina parempia vaihtoehtoja (Nielsen 2000, 22). Käyttöliittymän tulee mahdollistaa päätavoitteiden ja – toimintojen yksinkertainen, helppo ja nopea suorittaminen. Monimutkaisemmat ja pitkälle kehitetyt toiminnot eivät ole poissuljettuja, mutta pääpaino on yleisimmin suoritetuilla toiminnoilla ja tehtävillä. Nielsenin mukaan yksi www-suunnittelun tärkeimmistä periaatteista onkin pysyä pois käyttäjän tieltä ja auttaa käyttäjää saavuttamaan päämääränsä mahdollisimman nopeasti. (Nielsen 2000, 380.)

Yksi keino lisätä käytettävyyttä on käyttää elementtejä, jotka ovat jo entuudestaan käyttäjille tuttuja (Nielsen & Loranger 2006, 369). Esimerkkinä tutusta elementistä toimii sekä Windows- että muistakin ympäristöistä tuttu ruksi oikeassa ylänurkassa, jolla suljetaan sovellus tai sen osa. Tämä elementti lienee niin tuttu, ettei sen tilalle ole syytä keksiä uutta elementtiä tai toimintoa. Tyypillisten ja tuttujen elementtien uudelleensuunnittelu voi johtaa toimintojen suorittamisen hidastumiseen ja virhetilanteisiin, ja tätä kautta käyttäjien hämmentymiseen ja pahimmillaan sovelluksen käytön lopettamiseen (Nielsen & Loranger 2006, 372).

Niin nettisivujen kuin sovellustenkin käyttöliittymiin liittyvät pitkät dokumentit täytyy tarjota myös tulostettavana versiona. Tietokoneiden näytöt eivät ole paras mahdollinen työkalu suurten tekstimäärien lukemiseen. Näytöltä luetaan keskimäärin 25 prosenttia hitaammin kuin paperilta. Joissain tapauksissa paperitulosteita käytetään tietojen arkis-

tointiin. Nielsen suositteleekin, että pitkistä dokumenteista suunnitellaan omat versiot näytölle ja paperitulosteelle. (Nielsen 2000, 94, 101.) Työpöytäsovelluksessa tulostuksen kohteena voi olla joko käyttöohje tai koritiedot, joita käytetään työtehtävien suunnittelussa ja valmistelussa. Varsinaisesta arkistoinnista tuskin tulee olemaan tämän työpöytäsovelluksen kohdalla kyse.

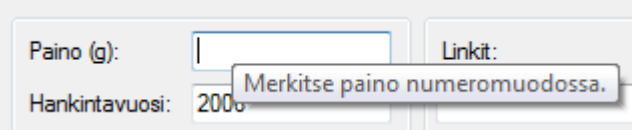
Käyttöliittymän tulisi olla aina niin helppokäyttöinen, ettei käyttäjän tarvitse turvautua erilliseen käyttöohjeeseen. Nielsen huomauttaa, että tarve käyttöohjeelle johtuu epäonnistuneesta suunnittelusta. Käyttöohjeita tarvitaan kuitenkin usein monimutkaisia tai pitkälle kehittyneitä toimintoja suoritettaessa. (Nielsen 2000, 129.) Ehdottoman tärkeä huomio käyttöohjeiden suhteen on, että käyttäjät eivät koskaan lue ohjeita vapaaehtoisesti, vaan ohjeeseen tartutaan vasta siinä vaiheessa, kun järjestelmän käytössä tulee vastaan ongelmia. Ohjetta luettaessa siitä silmäillään vain ne tiedot, jotka auttavat senhetkisen ongelman ratkaisuun. Tästä johtuen käyttöohjeen rakennetta määrittelevät tietyt perussäännöt. Tietoa pitää pystyä etsimään käyttöohjeesta mahdollisimman helposti ja nopeasti. Ohjeistuksessa tulee suosia esimerkkitapauksia, jotka helpottavat käyttäjän ongelmatilanteen ratkaisua paremmin kuin pelkkä yleisluontoinen selitys. Ohjeissa tulee käydä käyttötilanteita läpi vaihe vaiheelta, mutta kuitenkin lyhyesti ja ytimekkäästi. Mahdollisista vaikeaselkoisista käsitteistä ja termeistä kootaan erillinen sanasto. (Nielsen 2000, 131, 134.) Nielsen huomauttaa myös, että on parempi aliarvioida kuin yliarvioida käyttäjän tietoteknisiä taitoja. Käyttöliittymässä ja sen käytön ohjeistuksessa kannattaa välttää vaikeita teknisiä termejä. (Nielsen & Loranger 2006, 364.)

3.3. Työpöytäsovelluksen käytettävyys

Ohjelmiston käytettävyyttä ja käyttöliittymää voidaan tarkastella monen eri tahon luomien määritelmien mukaisesti. Opinnäytetyölleni asettamista tutkimuskysymyksistä yksi liittyy käytettävyyteen ja helppokäyttöisyyteen. Tärkeimpiä huomioita käytettävyyteen liittyen ovat mielestäni käyttöliittymän yhtenäisyys sekä tärkeiden toimintojen nopea ja helppo saavutettavuus ja suoritettavuus. Sovelluksen tai järjestelmän käyttö tulisi olla niin yksinkertaista, ettei käyttäjän tarvitse missään vaiheessa miettiä, miten erilaiset toiminnot suoritetaan (Krug 2006, 11).

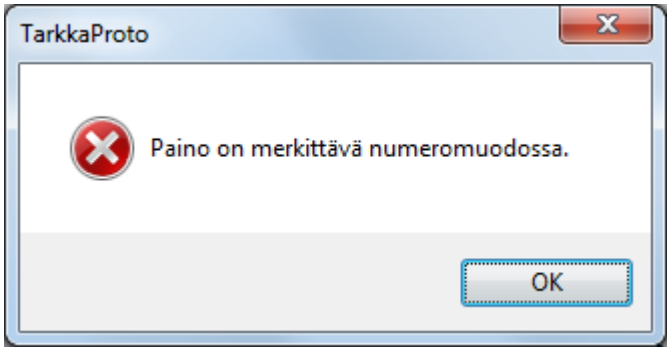
Tutkimusasetelmani mukaan haluan tutkia, millainen käyttöohje, oli se sitten kirjallinen tai sovellukseen lisätty, palvelee sovelluksen käyttäjää parhaiten. Käyttöohjeen tulisi mielestäni olla mahdollisimman selkeä, kattava ja ennen kaikkea lyhyt. Pitkät tekstimuotoiset käyttöohjeet eivät palvele käyttäjää, koska niistä tarvittavan tiedon hakeminen vaatii aikaa. Usein törmää siihen tilanteeseen, että käyttäjä yrittää selvittää tilanteen ensin itse. Kun omat kyvyt eivät riitä tilanteen ratkaisemiseen, tukeudutaan käyttöohjeeseen viimeisenä oljenkortena. Tässä vaiheessa käyttäjä on luultavimmin siinä mielentilassa, ettei hän jaksa eikä tahdo lukea pitkiä ohjeita.

Kun käyttöä ohjataan sovelluksessa mahdollisimman hyvin, säästyy käyttäjä varsinaisen käyttöohjeen lukemiselta. Sovelluksessa voidaan esimerkiksi antaa huomautuksia, mikäli kenttään voidaan syöttää vain tietynlaista tietoa. Tällainen huomautus on esitetty kuvassa 1.



Kuva 1. ToolTip-huomautus sovelluksessa

Mikäli käyttäjä yrittää syöttää kenttään muuta kuin numeromuotoista tietoa, huomauttaa sovellus tästä äänimerkin kanssa. Huomautus annetaan näyttöön ilmestyvällä MessageBox-elementillä, joten se tuskin jäänee käyttäjältä huomaamatta. Kuva 2 esittää MessageBox-huomauksen.



Kuva 2. MessageBox-huomautus sovelluksessa

Virheentarkistusta tulee vielä tarkentaa kehitettäessä prototyyppiä valmiiksi sovellukseksi. Esimerkiksi edellä mainittuun paino-kenttään liittyen prototyypistä puuttuu vielä

syötettyjen tietojen tarkastus tallennuksen yhteydessä. Tämä tarkoittaa sitä, että vaikka käyttäjälle annetaan huomautus paino-kentän vaatimasta numeromuotoisesta tiedosta, prototyyppi kuitenkin hyväksyy myös tekstimuotoisen tiedon tallennettaessa. Tämä heikentää käytettävyyttä, koska annetut ohjeistukset ovat ristiriidassa sovelluksen toiminnan kanssa. Tämä voi myös aiheuttaa virhetilanteita esimerkiksi laskettaessa koriin lisättyjen tarvikkeiden painoa.

Käytettävyyttä tulee miettiä myös käyttöohjetta suunniteltaessa ja tehtäessä. Ohjeen tulee olla kattava ja selkeä, mutta sen tulee tarjota oikeat ja toimivat vastaukset käyttäjän kohtaamiin ongelmiin helposti ja nopeasti. Käyttöohjetta voidaan ”keventää” havainnollistamalla ohjetta kuvilla.

Mielestäni tärkein huomio käytettävyydestä ja helppokäyttöisyydestä on, että mitä helpompi on oppia ja käyttää sovellusta, sitä vähemmän vaaditaan käyttöohjeelta. Suunniteltava sovellus tukee toimeksiantajayrityksen toimintaa sekä logistisesta että liiketoimintaprosessien näkökulmasta. Prototyypin avulla sovelluksen käytettävyyttä, käyttöliittymää ja toimintoja voidaan testata ja tarvittaessa karsia turhia ja lisätä puuttuvia ominaisuuksia. Kun sovellus ja sen käyttöliittymä ohjeineen vastaa sille määriteltyihin vaatimuksiin, tukee se työskentelyä ja tämän kautta koko yrityksen liiketoimintaa.

4 SOVELLUKSEN KEHITYS

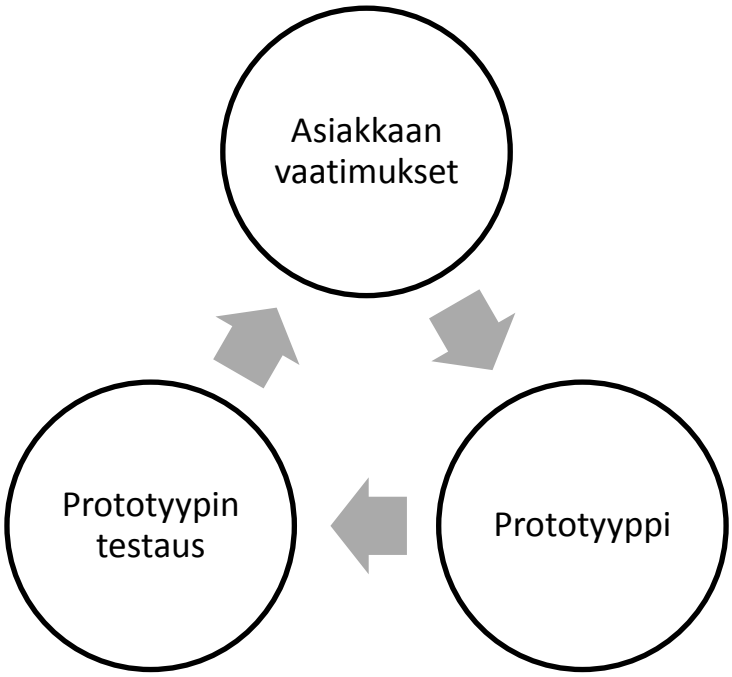
Työpöytäsovelluksen, kuten muidenkin ohjelmistojen, kehitys alkaa vaatimusten ja toiveiden määrittelystä. Määrittelyvaiheessa sovelluksesta ja tilaajan tarpeista ja toiveista muodostetaan mahdollisimman tarkka käsitys, johon tehtävät suunnitelmat perustuvat.

Opinnäytetyönä tehtävä sovellus koostuu Visual Studio-ohjelmointiympäristössä rakennetusta graafisesta käyttöliittymästä ja Access-tietokannasta. Access-tietokantaa voi tarvittaessa käyttää myös ilman sovellusta, kun taas sovellus ilman tietokantaa ei ole käyttökelpoinen.

Tässä luvussa käsittelen sovelluksen toteutukseen liittyviä työvaiheita ja -menetelmiä. Aluksi käyn läpi prototyypin kehitystä sekä työn etenemistä vesiputousmallin mukaisesti. Kerron myös suunnitteluvälineenä käytettävästä UML-mallinnuskielestä. Lopuksi käsittelen tutkimuskysymykseni mukaisesti tietokantaohjelman valintaa.

4.1 Prototyyppi

Opinnäytetyöni alkuperäisen suunnitelman mukaan tavoitteena oli tehdä täysin käyttövalmis sovellus. Prosessin aikana törmäsin kuitenkin ongelmiin toimeksiantajan vaatimusten suhteen. Keskusteltuani asiasta ohjaavan opettajani kanssa, muuttui tavoitteeni käyttövalmiista sovelluksesta prototyyppiin. Prototyyppi mahdollistaa suunniteltavan lopputuotteen piirteiden tai ominaisuuksien kokeilun ennen varsinaisen tuotteen rakentamista. Prototyyppilähestymistapa on erityisen toimiva uutta teknistä ratkaisua luotaessa sekä epäselvien asiakasvaatimusten selventämisessä. (Haikala & Märijärvi 2004, 42.) Kuvassa 3 on esitetty, millä tavoin asiakasvaatimusten ja prototyypin muotoutuminen voi edetä. Ensin määritetään asiakkaan vaatimukset, joiden pohjalta prototyyppi rakennetaan. Asiakas testaa prototyyppiä, jonka jälkeen vaatimuksia voidaan tarkentaa ja muuttaa. Uusien vaatimusten pohjalta voidaan rakentaa uusi prototyyppi tai kehittää edelleen ensimmäistä prototyyppiä.



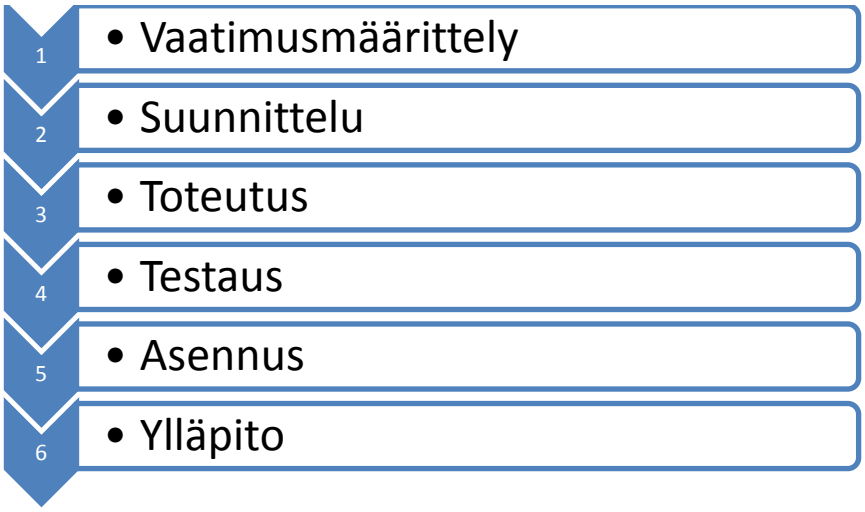
Kuva 3. Prototyypin kehitys (Taina 2000.)

Tuotettua prototyyppiä voidaan käyttää toteutettavan järjestelmän määrittelyyn tai prototyyppi itsessään voidaan kehittää valmiiksi järjestelmäksi (Haikala & Märijärvi 2004, 42). Prototyypistä voidaan myös kehittää toinen prototyyppi ensimmäisen version pohjalta saatujen uusien määritelmien perusteella (Taina 2000). Prototyyppi voi olla lähes identtinen varsinaisen lopputuotteen kanssa tai vaihtoehtoisesti hyvin puutteellinen esimerkiksi virhetarkistuksien, tietokannan ja opastuksen suhteen. Jälkimmäisellä voidaan varmistaa, että suunnitelmat sisältävät tarvittavat toiminnot ja käyttöliittymä on asiakkaalle sopiva. (Haikala & Märijärvi 2004, 42-43.) Prototyypin suunnittelussa ja toteutuksessa käytän samoja menetelmiä ja työvaiheita, kuin käyttövalmiin sovelluksenkin toteutuksessa.

4.2 Vesiputousmalli

Vesiputousmalli on yksi ohjelmiston elinkaarta tai kehitystyötä kuvaava vaihejakomalli. Vesiputousmallissa on tavallisimmin kuvattuna ainakin määrittely-, suunnittelu- ja toteutusvaiheet. Kuvassa 4 on esitetty yksi näkemys vesiputousmallin sisältämistä vaiheista. Vaiheisiin liittyy olennaisena osana laadunvarmistus erilaisin toimenpitein. Näitä toimenpiteitä, joita yleensä käytetään vaiheiden päättyessä, ovat esimerkiksi tarkastus, katselmus ja testaus. Erilaisilla testaus- ja katselmointitoimenpiteillä varmistetaan ja

seurataan projektin etenemistä suunnitelmien mukaisesti. (Haikala & Märijärvi 2004, 36-37.) Käytännössä pysyvä vaatimusmäärittely on mahdotonta tehdä, koska osa vaatimuksista selviää vasta projektin aikana tai jo määritellyt vaatimukset muuttuvat kesken projektin. Näin ollen ohjelmistokehitysprojekti ei ikinä etene täsmälleen vesiputousmallin mukaisesti, mutta sitä voidaan pitää projektin etenemisen ohjenuorana, jota noudatetaan mahdollisuuksien mukaan. (Haikala & Märijärvi 2004, 41.)



Kuva 4. Vesiputousmallin vaiheet

Vesiputousmallin mukainen työskentely alkaa vaatimusten määrittelystä. Vaatimusmäärittelyvaiheesta analysoiduista asiakasvaatimuksista johdetaan toteutettavaa järjestelmää määrittelevät ohjelmistovaatimukset. Määriteltävät vaatimukset koskevat muun muassa ohjelmiston toimintoja, rajoituksia, ominaisuuksia ja käyttöliittymää. (Haikala & Märijärvi 2004, 38-39.) Luotavan sovelluksen vaatimusmäärittelyä kokosin keskusteltuani toimeksiantajan kanssa sovellukseen kohdistuvista vaatimuksista ja toiveista. Kirjasin vaatimuksia ylös, ja tarvittaessa tarkensin niitä toimeksiantajan kanssa käydyissä keskusteluissa. Ensimmäinen merkittävä laadunvarmistustoimenpide oli määriteltyjen vaatimusten katselmointi yhdessä toimeksiantajan kanssa.

Vaatimusten määrittelyn jälkeen seuraa suunnitteluvaihe, jossa suunnitellaan ensimmäisessä vaiheessa kuvatut toiminnot. Voidaankin sanoa, että määrittelyvaihe vastaa kysymykseen ”mitä järjestelmä tekee?” ja suunnitteluvaihe ”miten järjestelmä tehtävänsä suorittaa?”. (Haikala & Märijärvi 2004, 40.) Suunnittelutyökaluna voidaan käyttää esimerkiksi erilaisia piirto-ohjelmia, kuten Microsoftin Visio-ohjelmaa (Haikala & Märijärvi 2004, 86). Suunnitteluvaiheesta siirrytään toteutusvaiheeseen, joka ohjelmistopro-

jektissa tarkoittaa ohjelmointia. Toteutusvaiheen tukena käytetään vaatimusmäärittelyä sekä suunnitteluvaiheessa tuotettuja dokumentteja.

Järjestelmän testaus on työkalu, jolla tutkitaan ja varmistetaan kokonaisvaltainen toiminnallisuus (McConnell 1998, 220) sekä mitataan ja parannetaan ohjelman laatua (Haikala & Märijärvi 2004, 284). Testauksessa huomion kohteena on järjestelmälle asetetut vaatimukset ja niiden riittävän laadukas täyttäminen (McConnell 1998, 220). Vesiputousmallin mukaisesti etenevässä projektissa järjestelmän testaus tapahtuu projektin loppuvaiheessa. Testausta on kuitenkin hyvä suorittaa jo aikaisemmassa vaiheessa järjestelmän eri toimintojen ja osien vaaditun toiminnallisuuden ja laadukkuuden varmistamiseksi.

Testausvaihe saattaa lohkaista projektin kokonaiskustannuksista suurimman osan sekä viedä huomattavan paljon aikaa (Haikala & Märijärvi 2004, 40). Yleensä testaus suunnitellaan vasta ohjelmiston ollessa jo lähes valmis. Tästä johtuukin puutteelliset esivalmistelut, lyhytaikainen testaus ja lopulta virheitä ja puutteita sisältävä julkaistu ohjelmisto (McConnell 1998, 220).

Testaukseen liittyviä työvaiheita ovat muun muassa testauksen suunnittelu, suorittaminen ja tulosten tarkastelu. Testaus mahdollistaa ohjelmiston virheiden osoittamisen. (Haikala & Märijärvi 2004, 283, 286.) Voidaankin sanoa, että testaus on onnistunut silloin, kun virheitä löytyy, koska virheetöntä ohjelmaa ei ole olemassakaan. Testaussuunnitelman laatiminen on tärkeää, koska sen avulla määritellään mitä ja miten testataan sekä millaisia tuloksia testistä odotetaan ja saadaan (Haikala & Märijärvi 2004, 299). Testitapaukset voidaan jakaa kahteen perusryhmään. Lasilaatikkotestaus tarkoittaa, että tietoa ohjelman toteutuksesta käytetään hyväksi testitapausten valinnassa. Mustalaatikkotestaus tarkoittaa päinvastaista, eli tietoa toteutuksesta ei käytetä vaan valinta tehdään testattavan kohteen spesifikaatioiden perusteella. Voidaan puhua myös harmaalaatikkotestauksesta, joka perustuu ohjelman toteutusperiaatteisiin. (Haikala & Märijärvi 2004, 291.) Prototyyppiin ei ole kohdistettu suunniteltua tai laajaa testausta, vaan sen sisältämien toimintojen testaus rajoittuu ohjelmointivaiheessa suoritettuun lyhyeen ja rajoitettuun testaukseen. Liitteessä 16 on kuitenkin esitetty dokumentti, jonka mukaisesti testausta ja testitapauksia voidaan suunnitella jatkokehityksen aikana.

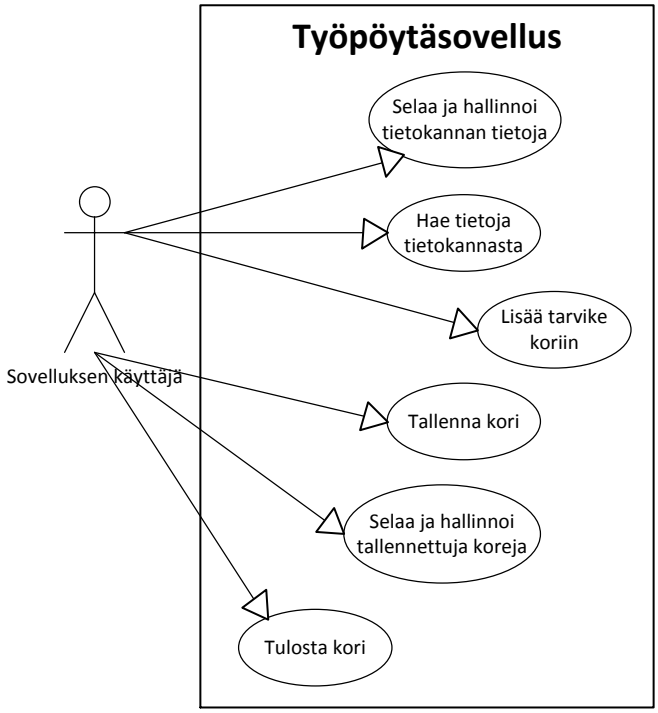
Ylläpitovaihe voi olla korjaavaa, adaptiivista tai täydentävää. Tehdyssä sovelluksessa havaittuja virheitä korjataan korjaavalla ylläpidolla. Adaptiivinen vaihe taas pitää sisällään sovelluksen muuttamista esimerkiksi muuttuneiden ympäristövaatimusten takia. Täydentävällä ylläpidolla voidaan lisätä sovelluksen toimivuutta tai muuten parantaa sitä. (Haikala & Märijärvi 2004, 41.)

Vesiputousmalli sopii opinnäytetyöhöni siinä mielessä erittäin hyvin, että vesiputousmallin mukaisessa työskentelyssä on perinteisesti panostettu suunnittelutyöhön. Opinnäytteessäni nimenomaan suunnittelen toteutettavaa sovellusta ja rakennan jatkokehityksen avuksi prototyypin.

4.3 UML-mallinnus

UMLeli Unified Modelling Language on graafinen mallinnuskieli, jota käytetään suunnitelmien kuvaamisessa (Fowler & Scott 2002, 2). UML-mallinnusta voidaan käyttää oliopohjaisten kielten, kuten Visual Basicin, määrittelyyn. UML:n eri osia, kuten kaavioita ja suhteita, käytetään dokumentointiin, määrittelyyn ja toteutukseen. (Kankaanpää 2010.)

UML-mallinnuskieltä voidaan käyttää kuvaamaan tietojärjestelmää eri näkökulmista. Keskeisimpiä UML-kaavioita on muun muassa käyttötapauskaavio (use case diagram). (Hovi & Huotari & Lahdenmäki 2003, 120-121.) Käyttötapauskaaviolla kuvataan järjestelmää käyttäjän näkökulmasta ja sitä hyödynnetään käyttötapausten määrittelyssä ja esittämisessä. Käyttötapauskaavioita voidaan hyödyntää erityisesti vaatimusmäärittelyvaiheessa sekä adaptiivisessa ylläpitovaiheessa. (Meriläinen 2008.) Kuvassa 5 on käyttötapauskaavio, jolla kuvataan tehtävän sovelluksen toimintoja sen käyttäjän näkökulmasta.



Kuva 5. Tarvikekannan hallintasovellusta kuvaava käyttötapauskaavio

Sekvenssikaavio (sequence diagram), joka tunnetaan myös nimellä viestiyhteyskaavio, kuvaa järjestelmässä tapahtuvia asioita aikajärjestyksessä (Fowler & Scott 2002, 66). Sekvenssikaaviota voidaan käyttää, mikäli halutaan tarkastella tiettyyn käyttötapaukseen sisältyvien olioiden käyttäytymistä (Fowler & Scott 2002, 68). Liitteessä 1 on sekvenssikaavio, joka kuvaa prototyypin sisältämien olioiden välisiä yhteyksiä.

4.4 Tietokantaohjelman valinta

Tietokantaohjelmien suhteen päädyin alustavasti kahteen vaihtoehtoon, joista toinen on ilmainen ja toinen maksullinen. Nämä vaihtoehdot olivat avoimen lähdekoodin tietokantaohjelma OpenOffice Base ja kaupallinen Microsoft Access.

OpenOffice on täysimittainen toimisto-ohjelmisto, joka perustuu avoimeen lähdekoodiin sekä avoimiin standardeihin. Avoin lähdekoodi tarkoittaa tietokoneohjelmistojen kehittämis- ja jakamistapaa. Avoimen lähdekoodien ohjelmia saa käyttää, kopioida, muuntaa ja jakaa vapaasti ilman lisenssejä. Avoimen lähdekoodin ohjelmistojen kehitykseen osallistuu maailmanlaajuisesti sekä yksityishenkilöitä että yrityksiä. Tämän takia ohjelmistovirheet löydetään ja korjataan nopeasti, ohjelmistot ovat korkealaatuisia,

yhteensopivia ja tietoturvaltaan tasokkaita. (Avoin lähdekoodi 2010.) OpenOffice toimisto-ohjelmisto on suunniteltu yhtenäiseksi kokonaisuudeksi. Avoimeen tuotekehitysprosessiin, joka on kestänyt jo yli kaksikymmentä vuotta, voi osallistua kuka tahansa. OpenOffice on saatavilla suomenkielisenä versiona. Kaikki sen osat ovat ilmaisia sekä vapaasti kopioitavia ja levitettäviä. OpenOffice-paketti sisältää muun muassa tekstinkäsittely-, taulukkolaskenta-, vektorigrafiikka-, esitysgrafiikka- ja tietokantaohjelmat. Ohjelmien toiminnallisuus vastaa kaupallisen Microsoftin Office-tuoteperheen ohjelmia. OpenOfficen ohjelmia levitetään LGPL-lisenssillä (Lesser General Public License), jonka ansiosta sekä yksityinen että kaupallinen käyttö on sallittua. (OpenOffice.org-toimisto-ohjelmisto 2010.)

Microsoftin Access-tietokantaohjelmisto on osa suurempaa Microsoft Office-tuoteperhettä. Toisin kuin OpenOffice, Access on maksullinen. Accessin voi hankkia osana Office-pakettia tai yksittäisenä ohjelmistona. Base ja Access sisältävät samantyyppisiä toimintoja ja esimerkiksi Access-tietokantaa voidaan käsitellä myös Base-tietokantaohjelmalla. Molempien ohjelmien hyvänä puolena on se, että tarvittaessa tietokanta on helppo siirtää ja käyttää sellaisenaankin. Sekä Base että Access sisältävät graafisen käyttöliittymän ja useita ohjattuja toimintoja, kuten tietokantakyselyiden ja raporttien muodostamisen. Basen ja Accessin käyttö sellaisenaan vaatii kuitenkin käyttäjältä perustietojen hallitsemista tietokantojen suunnittelun ja toteuttamisen osalta.

Lopullinen tietokantaohjelman valinta kohdistui Microsoft Accessiin. Access-tietokantaan on helppo siirtää tietoja Excel-taulukosta, jossa tarviketiedot tällä hetkellä ovat. Päätökseen vaikutti myös toimeksiantajan toiveet ja yrityksen tämän hetkinen tilanne. Toimeksiantaja piti parhaana ratkaisuna saman tuoteperheen ohjelmien käyttöä. Tämän lisäksi yrityksessä ajankohtaista oli uuden tietokoneen hankinta Windows 7 -käyttöjärjestelmän ja Microsoftin ohjelmistojen kanssa. Toimeksiantaja suunnittelikin hankkivansa konepakettiin myös Microsoft Access-tietokantaohjelman.

5 RATKAISUN TOTEUTUS

Luotava sovellus muodostuu karkeasti sanottuna kahdesta osasta: tietokannasta ja käyttöliittymästä. Käyttäjälle näkyy vain käyttöliittymä, jonka kautta hän voi käsitellä sovelluksen pohjana olevaa tietokantaa ja sen sisältämiä tietoja.

Järjestetty kokoelma toisiinsa liittyviä tietoja muodostaa tietokannan. Tietokannan sisältämiä tietoja pystytään hallinnoimaan ja käsittelemään erilaisilla tietokantaohjelmilla, kuten Microsoftin Accessilla tai avoimen lähdekoodin ohjelmalla OpenOffice Base. (Hyppönen & Ojala & Joutsi 2007, 290.) Nykyaikaisten sovellusten perustana on useimmiten tietokanta, vaikka käyttäjille näkyy vain käyttöliittymä ja erilaiset raportit (Hovi ym. 2003, 20).

Käyttöliittymä on kokoelma komentoja tai valikoita, joiden kautta käyttäjä kommunikoi tietokonesovelluksen kanssa. Käyttöliittymä on yksi tärkeimmistä sovelluksen osista, koska sen kautta määrittyy sovelluksen käytettävyys. Nykyisin sovellusten käyttöliittymät ovat pääasiallisesti graafisia. (User Interface 2010.) Graafista käyttöliittymää käytettäessä ei tarvitse muistaa tai opetella monimutkaisia komentokieliä, koska komentoja ei syötetä tekstimuotoisena. Erilaiset graafiset käyttöliittymät koostuvat perusosista, kuten ikkunoista, ikoneista ja valikoista. Vaikka ensimmäinen graafinen käyttöliittymä on tehty jo 1970-luvulla, kasvoi niiden suosio vasta 1980-luvulla Applen Macintosh-koneiden myötä. Graafisten käyttöliittymien käyttöönottoa hidasti vaadittavien prosessitehojen ja korkealaatuisten näyttöjen hintavuus. (Graphical User Interface 2010.)

Tässä kappaleessa käyn läpi tietokannan ja käyttöliittymän suunnittelun periaatteita ja valitsemiani toteutustapoja.

5.1 Tietokannan suunnittelu

Työpöytäsovelluksen prototyypin pohjalle on tehty tietokanta, joka mahdollistaa sovelluksen erilaisten toimintojen testaamisen ja suunnittelemisen. Voidaankin puhua testaustietokannasta, jota ei kuitenkaan tule käyttää varsinaisen sovelluksen pohjalla. Tässä luvussa käsittelen tietokannan suunnittelun periaatteita ja kerron prototyypin pohjalle

tehdystä tietokannasta. Luvussa 6.1 käyn läpi tarkemmin asioita, joita tulee tietokannan osalta ottaa huomioon, kun prototyypistä kehitetään valmis sovellus.

Tietokanta muodostuu tauluista (table), joihin kaikki tietokannan tiedot tallennetaan. Taulut puolestaan muodostuvat riveistä ja sarakkeista. Taulujen vaakasuora rivi, eli tietue, sisältää tietoa vain yhdestä yksittäisestä asiasta. Pystysuorat sarakkeet, eli kentät, sisältävät vain yhdenlaista tietoa. Kenttiin tulevan tiedon tyyppi määritellään tietokantaohjelmassa. Kentän tietotyyppi voi olla esimerkiksi luku- tai tekstimuotoista. (Hyppönen ym. 2007, 290.)

Taulukoiden tietueet tulee yksilöidä tai erottaa toisistaan. Erotus tapahtuu perusavaimella (primary key), jonka sisältö ei voi koskaan olla tyhjä ja joka on jokaisessa tietueessa erilainen. Tietoja voidaan hakea kaikista tietokannan tauluista määrittämällä tauluille keskinäiset yhteydet. Keskinäinen yhteys muodostetaan viiteavaimella (foreign key). Viiteavaimella tarkoitetaan tauluun lisättyä toisen taulun avainkenttää. (Hyppönen ym. 2007, 291.)

Kuvassa 6 on esimerkki työpöytäsovelluksen prototyypin yhdestä taulusta. Tietue koostuu yhden kameran eri tiedoista, kuten valmistajasta, hinnasta ja tyypistä. Kentät sisältävät samanlaisia tietoja, esimerkiksi valmistaja-kentässä on aina vain valmistajan nimi. Perusavain kuvan esimerkissä on jokaiselle kameralle annettu id-numero.

Kentän nimi	Tietotyyppi
runkoID	Laskuri
runkoValmistaja	Teksti
runkoTyyppi	Teksti
runkoPaino	Luku
runkoValmistusnro	Teksti
runkoValmistusvuosi	Teksti
runkoHankintavuosi	Teksti
runkoHintaEur	Teksti
runkoLisatieto	Memo
runkoLinkki1	Teksti
runkoLinkki2	Teksti
runkoLinkki3	Teksti
runkoKuva	Teksti

Kuva 6. Tarviketietokannan rungot-taulun rakenne

Relaatiomallin mukainen tietokanta koostuu useasta eri taulusta. Tietokantaan lisättävät tiedot ryhmitellään aiheittain, jonka jälkeen jokaiselle aiheryhmälle luodaan oma taulu. Luomalla taulujen välille looginen yhteys, voidaan taulujen tietoja käsitellä samanaikaisesti ja tauluista muodostuu näin yksi kokonaisuus. (Hyppönen ym. 2007, 292.)

Taulujen välillä voi olla erilaisia yhteystyyppejä. Näitä yhteystyyppejä ovat yksi-moneen-, yksi-yhteen- ja monta-moneen-yhteydet. Yksi-moneen-yhteys on yleisimmin käytetty yhteystyyppi. Tässä yhteydessä ensisijaisen taulun yksi tietue voi viitata useaan toisen taulun tietueeseen, ja toisen taulun tietue viittaa aina vain yhteen ensisijaisen taulun tietueeseen. (Hyppönen ym. 2007, 322.) Esimerkiksi yksi valmistaja on voinut valmistaa monta linssiä, mutta yhdellä linssillä voi olla vain yksi valmistaja. Harvinaisempi yksi-yhteen-yhteystyyppiä voidaan käyttää, jos tauluun näyttäisi jäävän suuri määrä tyhjiä kenttiä. Yksi-yhteen-yhteystyyppi tarkoittaa yleensä sitä, että taulujen tiedot voidaan sijoittaa samaan tauluun. (Hyppönen ym. 2007, 323.) Luotavan työpöytäsovelluksen kohdalla tälle yhteystyypille ei ole tarvetta. Monta-moneen-yhteystyyppi vaatii niin sanotun liittämistaulun. Liittämistauluun muodostetaan kahdesta tarvittavasta taulusta yksi-moneen-yhteys lisäämällä niiden perusavaimet liittämistaulun viiteavaimiksi. (Hyppönen ym. 2007, 323.) Prototyypissä monta-moneen-yhteys löytyy koriin liittyvistä tauluista. Korin sisältö-tauluun yhdistetään sekä kori- että tarviketaulut id-numeroja käyttäen.

Tietokannan hyvä suunnittelu ja rakentaminen ennaltaehkäisevät hankalien ja sovellusohjelmalla paikattavien tietorakenteiden muodostumista sekä helpottaa ohjelmointia. Tietokannan suunnittelu pitää sisällään muun muassa vaatimusten määrittelyä, tietokannan mallinnusta ja fyysistä suunnittelua. Tietokannan mallinnus tarkoittaa pääasiallisesti jonkun kuvaustekniikan käyttöä tietokannan kuvaamisessa. Nykyaikaiset tietokanta- ja sovelluskehitystuotteet mahdollistavat taulujen helpon toteutuksen. Siitä huolimatta on kattavan suunnitelman tekeminen sitä tärkeämpää, mitä monimutkaisempi ja laajempi sovelluskokonaisuus on kyseessä. (Hovi ym. 2003, 20.)

Prototyypin pohjalle tulevaan tietokantaan tiedot ryhmitellään tarvikeryhmien mukaisesti. Linssit, kameroiden rungot ja runkojen osat tulevat kaikki omiin tauluihinsa, koska niillä kaikilla on omat yksilölliset ominaisuudet ja niille kirjattavat tiedot vaihtelevat tarviketyypin mukaan. Tarvikekori muodostuu kahdesta eri taulusta. Toiseen tauluun tallennetaan korin yksilöllinen tunnus ja korille annettu nimi. Koriin lisätyt tarvikkeet

tallennetaan omaan tauluunsa. Yhteensopivuustietoa tarvitaan vain työpöytäsovelluksen kori-toiminnossa. Koska tarvikkeiden yhteensopivuus määritellään lähes täysin liitteessä 2 kuvatun kaavan mukaisesti, ei yhteensopivuustietoja lisätä erikseen tietokannan tauluihin. Prototyypissä tarvikkeiden yhteensopivuuden tarkistaminen tapahtuu sovelluksen koodissa.

Tietokannan suunnittelussa voidaan edetä vaiheittain alkaen käsite- ja tarveanalyysistä, joiden avulla voidaan siirtyä normalisointiin, toteutukseen ja lopulta suorituskyvyn viritykseen. Tietokannan suunnitteluvaiheet eivät etene vesiputousmallin mukaisesti, vaan yleensä iteratiivisesti eli vaiheet toistuvat suunnittelun edetessä. Käsiteanalyysivaiheessa tietokanta suunnitellaan ja mallinnetaan alustavasti. Tarveanalyysissä tarkennetaan ja täydennetään tehtyä käsiteanalyysiä sovelluksen tietotarpeiden, kuten käyttöliittymän, perusteella. Normalisoinnilla tietokannasta ja sen rakenteesta pyritään karsimaan tietojen toistuvuus. Tietokannan toteutuksen jälkeen suorituskykyä voidaan virittää indeksoimalla eli tietokantahakuja nopeuttavien tietotyyppien määrittämisellä. (Hovi ym. 2003, 24-25, 337.)

Tietokannan suunnittelu työpöytäsovelluksen kohdalla lähti tietomäärän ja -laadun kartoittamisella. Toimeksiantaja oli tallentanut tarviketietoja Excel-taulukkoon, joten tietokantaan tulevien tietojen määrittäminen oli tätä kautta helppoa. Määrittämisen kautta ilmeni, että tarvikekategoriat voidaan jakaa pää- ja alakategorioihin. Nämä kategoriat on esitetty liitteessä 3. Pääkategorioita on yhteensä kahdeksan ja niihin liittyviä alakategorioita yhteensä kaksikymmentäviisi kappaletta. Pääkategorioihin on määritelty joukko tarvikkeita, joita yhdistää joku ominaisuus tai määre, esimerkiksi Large Format-tarvikkeet tai Enlargement- eli suurennuskonetarvikkeet. Alakategorioissa on jaettu pääkategoriaan liittyvät tarvikkeet edelleen omiin ryhmiinsä, esimerkiksi linsseihin, runkoihin ja runkojen osiin.

Kategorioiden määrittämisen jälkeen tarkastelun kohteeksi otettiin jokaisen alakategorian eli tarvikeryhmän sisältämät tiedot. Toimeksiantaja määritteli, mitä tietoja hän haluaa käyttää ja mitkä voidaan karsia pois. Tarviketietoja määriteltäessä havaitsin, että mitään alakategorioita ei voida suoraan yhdistää, koska jokaisen alakategorian tiedot olivat erilaiset. Tiedoissa oli kuitenkin myös osittain yhteneväisyyksiä. Liitteessä 4 on esimerkkinä kirjattu kahden eri alakategorian sisältämät tiedot. Liitteestä käy ilmi, miltä osin tiedot ovat yhteneväisiä ja miltä osin eroavia.

Normalisoinnilla pyritään tietojen toistuvuuden minimointiin, tehokkaaseen tietokannan päivitykseen sekä yhdenmukaisuuteen ja muutosjoustavuuteen. Normalisointia voidaan tarkastella kolmen normaalimuodon (normal form) kautta ja onkin hyvä pyrkiä kolmanteen normaalimuotoon. Ensimmäisessä normaalimuodossa poistetaan toistuvat tietoryhmät ja moniarvoiset kentät. Toinen normaalimuoto on jo haasteellisempi. Siinä tarkastellaan tietokannan sisältämiä funktionaalisia riippuvaisuuksia. Toisin sanoen, mikäli taulu sisältää moniosaisen perusavaimen, tulee kaikkien sarakkeiden olla funktionaalisesti riippuvia koko perusavaimesta. Kolmas normaalimuoto jatkaa edellisen jalanjäljillä. Kolmannessa normaalimuodossa varmistetaan, että jokainen sarake on funktionaalisesti riippuvainen vain perusavaimesta. (Hovi ym. 2003, 86, 88, 90-91, 93.)

Normalisoinnin vastakohtana on denormalisointi, joka tarkoittaa tietojen tahallista toistamista. Tämän seurauksena tietokannan taulujen määrä vähenee, tietojen lukeminen nopeutuu ja tietokantakyselyt yksinkertaistuvat taululiitosten vähentyessä. (Hovi ym. 2003, 95.) Yleisesti suositellaan normalisoimaan tietokannat, mutta on myös esitetty näkemyksiä, joiden mukaan liika normalisointi ei ole kannattavaa. Normalisointia suositellaan tietokannan päivitysprosessin helpottamisen kannalta, mutta se voi aiheuttaa ongelmia tietokantakyselyissä. Usein tietokantaa joudutaankin denormalisoimaan, jotta voidaan täyttää siihen kohdistuneet toiminto- ja muut vaatimukset. (Agarwal & Huddleston 2008, 33.)

Koska prototyypin ohjelmoinnin aikana törmäsin ongelmiin tietokantayhteyden ja relaatiotietokannan käytön kanssa, päätin pelkistää rakenteen mahdollisimman yksinkertaiseen muotoon. Tämä tarkoitti myös tehdyn osittaisen normalisoinnin poistamista. Denormalisoin prototyypin tietokannan niin, ettei se täytä edes ensimmäisen normaalimuodon vaatimuksia. Alun perin olin luonut tietokantaan erilliset taulut tarvikkeiden valmistajille ja tyypeille, koska nämä tiedot toistuvat usein. Liitteessä 5 esitetään prototyypin pohjalla käyttämäni tietokannan rakenne ja liitteessä 6 taas on esimerkki varsinaisen sovelluksen pohjalle tulevan tietokannan rakenteesta.

5.2 Käyttöliittymä ja GUIDe-prosessimalli

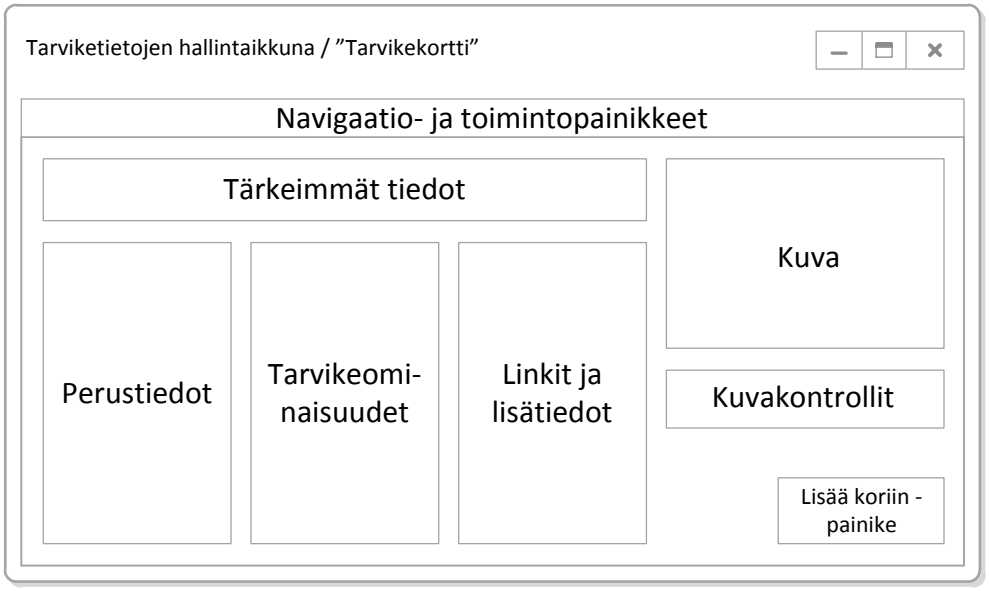
GUIDe-prosessimallin mukaisessa suunnittelussa järjestelmä ja siihen liittyvä käyttöliittymä suunnitellaan käyttötilanteiden ja tavoitteiden pohjalta. Tällainen suunnittelu edel-

lyttää käyttäjien tavoitteiden ja työnkulkujen selvittämistä sekä mahdollisesti tarvittavaa suoraviivaistamista heti projektin alussa. Selvitystyön perusteella käyttöliittymäsuunnittelija työstää tavoitteita ja työnkulkuja mukailevat ja tukevat käyttöliittymäratkaisut. Ratkaisuihin kuuluu olennaisena osana käyttöliittymän toimintojen yksityiskohtainen kuvaaminen. Ratkaisut esitetään kuvasarjoina, joista käy ilmi käyttöliittymän tarjoamat toiminnot, toimintojen käyttötavat, tietosisältö ja sen esittäminen. Näin käyttöliittymän toiminnallisuus ja käytettävyys voidaan testata ja puutteet ja virheet korjata ennen varsinaisen toteutuksen aloittamista. Prosessimallin nimi GUIDe tuleekin sanoista Goals, User Interface Design ja Implementation, eli tavoitteet, käyttöliittymäsuunnittelu ja toteutus. (Laakso & Laakso 2004.)

Opinnäytetyössä käyttämäni vesiputousmalli ei sisällä erillistä vaihetta käyttöliittymän suunnittelua varten. Tämän takia tiukasti vaiheistettua vesiputousmallia noudatettaessa lopullisen käyttöliittymän ulkonäkö, toiminnot ja käytettävyys näkyvätkin vasta projektin loppuvaiheessa. Tästä voi seurata suuria ongelmia nimenomaan järjestelmän tai sovelluksen käytettävyyden kanssa. Suunniteltavan järjestelmän ongelmia ja riskejä kartoitetaan vesiputousmallin alkuvaiheessa, ja näin tulisi tehdä myös käyttöliittymän suhteen. (Laakso & Laakso 2004.)

Heti projektin alussa suunnittelin käyttöliittymän toimeksiantajan esittämien toiveiden ja vaatimusten pohjalta. Käyttöliittymäsuunnitelman tein Visual Studion editorissa, koska sen kautta erilaisten käyttöliittymäkomponenttien lisääminen käyttöliittymään helppoa ja nopeaa. Visual Studion editorin käyttö mahdollisti myös käyttöliittymäsuunnitelman pikaisen muokkaamisen palaverin aikana. Alustavasta käyttöliittymäsuunnitelmasta keskustelin toimeksiantajan kanssa. Tämän keskustelun pohjalta käyttöliittymään tuli useita muutoksia ja toimeksiantajan toiveet ja vaatimukset tarkentuivat.

Työpöytäsovelluksen käyttöliittymä rakennetaan käyttämällä Visual Studion valmiita objekteja, kuten tekstikenttiä, kuvaruutuja ja painikkeita. Visual Studion sisältämiä kontrolleja, joita on käytetty käyttöliittymää rakennettaessa, on esitetty liitteessä 7. Käyttöliittymässä pyritään mukailemaan Windows-maailmasta tuttujen ohjelmistojen käyttöliittymiä käytön oppimisen helpottamiseksi. Kuva 7 esittää tarviketietojen hallintaikkunan käyttöliittymän rakenteen pelkistettynä.



Kuva 7. Tarviketietojen hallintaikkunan eli tarvikekortin käyttöliittymän rakenne

Käyttöliittymän päätoiminnot, eli tietojen selaus, korien hallinta ja tietojen haku, sijoitetaan eri lomakkeille. Lomakkeet helpottavat usean eri tiedon selaamista samanaikaisesti. Linssit, rungot ja runkojen osat ovat kaikki sijoitettu omalle lomakkeelle, ja näitä lomakkeita voidaankin käyttää yhtäaikaisesti. Koska erilaiset tarvikkeet kuitenkin liittyvät toisiinsa, on järkevää, että niiden tietoja pystytään tarvittaessa selaamaan vierekkäin asetettavilta lomakeikkunoilta. Aluksi olin epäileväinen useiden erillisten lomakeikkunoiden suhteen, mutta tätä samaa käyttöliittymä mallia noudattaa esimerkiksi avoimen lähdekoodin kuvankäsittelyohjelma GIMP. GIMPissä erilaiset työkalu- ja muut hallintapaneelit avautuvat kaikki omiin ikkunoihinsa, ja avoinna voikin olla useita hallintaikkunoita samanaikaisesti.

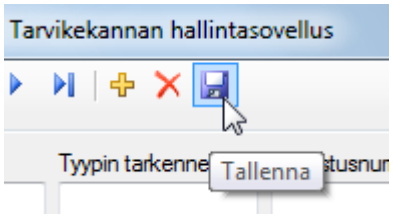
Yhtenäinen ulkoasu sovellukselle rakennetaan käyttämällä samanlaisia elementtejä ja samanlaista rakennetta mahdollisimman laajasti. Siirtymien, värityksen ja sisällön sijoittaminen yhtenäistäminen helpottaa sovelluksen rakenteen hahmottamista ja käyttöä. Erilaisten elementtien järjestäminen rajattuun tilaan on sommittelua. Sommittelun avulla voidaan ohjata käyttäjän katsetta, herättää mielenkiintoa ja painottaa elementtien tärkeysjärjestystä. Sommittelua tai elementtien järjestämistä suunniteltaessa tulee miettiä käyttäjän tarpeita ja suoritettavien toimintojen tavoitteita (Korpela & Linjama 2003, 363, 369, 371).

Työpöytäsovelluksessa ja tietokannassa tarviketiedot vaihtelevat pääryhmän mukaisesti. Linssit-kategoriassa on eniten ja osat-kategoriassa vähiten tietoja. Eri tarvikekategorioi-

den tiedot pyritään kuitenkin järjestämään samantyyppisesti. Tietojen looginen ja samantyyppinen järjestys helpottaa sovelluksen käyttöä ja käytön opettelua. Länsimaissa lukusuunta on vasemmalta oikealle ja ylhäältä alas, ja tämä on huomioitu myös sovelluksen käyttöliittymässä. Ensimmäisenä ylhäällä vaakatasossa on näkyvissä tarviketiedoista tärkeimmät. Toimeksiantajan toiveena oli, että tärkeimmät tiedot ovat helposti näkyvillä ja selkeästi erotettuna muista toisarvoisista tiedoista. Koska tiedot on sijoitettu GroupBox-ryhmäobjektiin, ovat ne selkeästi yksi tietoryhmä.

Siirryttäessä lomakkeella alaspäin, näkyvät seuraavaksi tarvikkeiden muut, myös GroupBox-objektilla ryhmitellyt tiedot. Vasemmalta päin katsottuna ensimmäinen tietoryhmä on perustiedot, jota seuraa tarvikkeen ominaisuudet, lisätiedot ja lopuksi kuva.

Tietojen selaus tapahtuu Visual Studion automaattisesti luomalla navigaatiopalkilla, joka löytyy lomakeikkunan yläreunasta. Navigaatiopalkin painikkeet koskevat koko tietuetta. Kuvan lisäys- ja poisto-painikkeet onkin sijoitettu kuvaruudun alle, koska ne koskevat vain kuvatietoa. Tietueen tietoja voidaan selata eteen- ja taaksepäin sekä siirtymällä suoraan ensimmäiseen tai viimeiseen tietueeseen. Navigaatiopalkista löytyy myös lisää uusi -, poista- ja tallenna-painikkeet. Navigaatiopalkin painikkeet ovat kuvamuodossa, ja ne noudattavatkin universaaleja selaus- ja sovelluspainikemalleja. Samantyyppiset selauspainikkeet voi löytää esimerkiksi levysoittimista ja videonauhureista ja sovelluspainikkeet erilaisista toimisto-ohjelmistoista. Kuvapainikkeita tulee käyttää harkiten. Kuvapainikkeiden vaarana on, että käyttäjä ei ymmärrä niiden merkitystä. Toisaalta kuvapainikkeet hyvin käytettynä ja tarkoin valikoituina tekevät sovelluksesta selkeän ja visuaalisesti miellyttävämmän. Kuvapainikkeisiin voidaan lisätä tekstimuotoinen ohjeistus esimerkiksi ToolTip-kontrolliin, joka näkyy hetken ajan käyttäjän viedessä kursorin painikkeen päälle. Tällainen ohjeistus on esitetty kuvassa 8.



Kuva 8. ToolTip-ohjeistus kuvapainikkeessa

Tarviketiedot näytetään lomakkeella pääosin TextBox-tekstikenttäkontrollissa. TextBox-kontrolli mahdollistaa tietojen esittämisen sekä helpon päivittämisen. Tietoja päivi-

tettäessä TextBox-kontrolliin kirjoitetaan uusi tieto, joka tallentuu tietokantaan Tallenna-painikketta klikkaamalla. Tarviketiedot olisi voitu esittää myös Label-etikettikontrollissa, mutta tämä olisi tehnyt tietojen päivityksen hieman monimutkaisemmaksi. Käyttäjä ei voi kirjoittaa suoraan Label-kontrolliin mitään, vaan päivitettäessä tieto olisi pitänyt kirjoittaa esimerkiksi TextBox-kontrolliin, jonka jälkeen se olisi näytetty Label-kontrollissa.

Käyttäjä hakee tietueeseen lisättävän kuvan valintaikkunaohjausobjektia käyttäen. Kun kuva on valittu, näytetään sen tiedostopolku TextBox-kontrollissa, josta se Tallenna-painiketta klikattaessa tallentuu tietokantaan. Kuvan valinta tapahtuu näin siksi, että tällä minimoidaan virheellisten kuvatietojen syöttäminen tietokantaan. Valintaikkunaohjausobjekti myös helpottaa kuvan lisäämistä siten, ettei käyttäjän tarvitse itse kirjoittaa tai muistaa lisäämiensä kuvien tiedostopolkuja.

Prototyypissä on huomioitu erilaisia mahdollisia virhetilanteita. Mahdollisista virhetilanteista pyritään ilmoittamaan käyttäjälle. Yksi tällainen virhetilanne on esimerkiksi kuvan näyttäminen tarvikekortissa. Tietokantaan tallennetaan lisätyn tarvikekuvan tiedostopolku, joka voi olla esimerkiksi ”C:\Users\Public\Pictures\Tarvikekuva1.jpg”. Virhetilanne voikin syntyä, mikäli kyseisellä tiedostopolulla ei löydäkään kuvaa. Tämä on huomioitu sovelluskoodissa, jossa testataan, voidaanko kuva määritellystä polusta näyttää. Jos tämä ei onnistu, annetaan käyttäjälle huomautus. Sovellus myös tarkastaa kuvaa lisättäessä, onko lisättävä tiedosto kuvatiedosto. Hyväksyttyjä tiedostomuotoja ovat .jpg- ja .jpeg-tiedostot. Mikäli käyttäjä lisää muun muotoisen tiedoston, annetaan tästä huomautus ja ohjeistetaan lisäämään kuvatiedosto. Tämä toiminto on kuvattu vuokaaviolla liitteessä 8. Koodiesimerkki on esitetty liitteessä 9.

6 PROTOTYYPISTÄ VALMIIKSI SOVELLUKSEKSI

Vaikka kehittämäni prototyyppi on toimiva sovellus sellaisenaan, ei sitä kannata ottaa käyttöön sen sisältämien puutteiden takia. Opinnäytetyöprosessin tiukka aikataulu ei mahdollistanut kohtaamieni ongelmatilanteiden riittävää ratkaisua. Prototyypissä usea ominaisuus ja toiminto kärsivät esimerkiksi ohjelmisto-ongelmien ratkaisemattomuudesta. Tämän luvun tarkoituksena on esitellä prototyypin jatkokehityksen kannalta tärkeitä puutteita ja huomioita. Aluksi keskityn tietokannan hyödyntämiseen sovelluksessa. Käsittelen myös sovelluksen käytettävyyttä ja virhetilanteiden ja -toimintojen estämistä.

6.1 Tietokannan hyödyntäminen ja kehitys

Tietokannasta muodostui sovelluksen prototyypin suurin kompastuskivi ja ongelma. Alkuperäisen tietoryhmitelmän mukaan tietokanta olisi sisältänyt kaksikymmentäviisi tarviketaulua, jotka puolestaan olisivat sisältäneet yli viisikymmentä osittain samaa tietoa. Esimerkiksi linsseihin liittyviä tauluja olisi ollut yli viisi. Tauluja ei kuitenkaan olisi voitu yhdistää yhdeksi suureksi linssitauluksi, koska tallennettavat tiedot erilaisten linssien välillä vaihtelivat. Kokonaisuudessaan tiedot olivat hyvin pirstaleiset ja jossain määrin turhaan toistuvat. Liitteessä 4 vertaillaan kahden linssiryhmän tietoja. Osa tiedoista, kuten tarvikkeen tyyppi ja valmistusnumero, toistuvat lähes kaikissa tauluissa. Osa tiedoista taas liittyy vain yhteen tiettyyn ryhmään. Tämä taas vaikeuttaa tietokannan normalisointia.

Tietokanta normalisoidaan tietojen toistuvuuden välttämiseksi. Toisaalta tietokanta voidaan myös denormalisoida, eli jätetään tietokantaan toistuvia tietoja tahallisesti. Alkuperäisen suunnitelman mukaan pelkkiä tarviketauluja olisi tietokannassa ollut jo yli kaksikymmentä kappaletta. Jos näitä tauluja lähdetään normalisoimaan, kasvaa taulumäärä huomattavasti. Haasteellista olisi myös se, miten tietokanta normalisoidaan. Mikäli tietokannan kahdestakymmenestäviidestä taulusta ainakin viisi sisältää kentän ”valmistaja”, tulee miettiä luodaanko yksi suuri valmistajat-taulu vai jokaiselle tarvikeryhmälle oma valmistaja-taulu. Tässä ongelmana on jälleen tietojen osittainen vaihtuvuus ja samankaltaisuus. Yksi valmistaja on voinut valmistaa eri ryhmien tai taulujen sisältämiä tarvikkeita. Näin ollen olisi järkevää yhdistää valmistajat yhteen tauluun.

Mutta toisaalta yksi valmistaja on voinut valmistaa vain yhden ryhmän tarvikkeita. Valmistajat-tauluun voidaan lisätä tarvikeryhmäsarakkeet, joihin merkitään esimerkiksi kyllä/ei vastauksella valmistajaan liittyvät tarvikeryhmät. Valmistajat-tieto ei kuitenkaan ole ainnoa tietoryhmä, joka kannattaa normalisoida omaan tauluun. Koska omiin tauluihinsa normalisoitavia tietoja on useita, kohdistuu huomio seuraavaksi tietokantakyselyihin. Mitä monimutkaisempia tietokantakyselyitä muodostetaan, sitä suurempi on virheen mahdollisuus ja sitä pitempään kyselyn suorittamiseen menee. Useissa lähteissä onkin varoitettu liiasta tietokannan normalisoinnista. ItPro.fi-sivustolla huomautetaan normalisointia olevan mahdollisesti liikaa, mikäli kyselyissä liitoksia (JOIN) on lähes aina neljä tai enemmän (Kyselyiden optimointi 2007). Sekä normalisoinnin että denormalisoinnin tarkoituksena on riittävästi parantaa ja optimoida tietokannan rakennetta, jotta rakenne tukee tietokannan varsinaista käyttötarkoitusta (Agarwal & Huddleston 2008, 33).

Loin prototyypin pohjalle relaatiotietokannan, jota en kuitenkaan pystynyt sovelluksessa täysin hyödyntämään. Visual Studio-ohjelmointiympäristössä ongelmana oli tietokantayhteyden osittainen toimimattomuus. Tietokantayhteys toimi vain yhteen suuntaan ja silloinkin väliaikaisesti. Kehitysvaiheessa tietokannan tiedot näkyivät lomakkeella oikein, tietokantahaku toimi moitteetta ja tietojen lisäys ja päivitys onnistui. Kun Visual Studio sulkemisen jälkeen päivitetyt tiedot kuitenkin hävisivät tietokannasta. Tähän ongelmaan hain ratkaisua, mutten sitä löytänyt. Jos aikataulu olisi sallinut, olisin tutkinut ongelmaa kunnes olisin löytänyt ratkaisun. Koska tietokantayhteys kuitenkin toimi sovelluksen julkaisuversiossa, jätin ongelman syvällisemmän tutkimisen prototyypin seuraavaan kehitysvaiheeseen.

Prototyypin tietokanta sisältää kolme tarviketaulua: linssit, rungot ja runkojen osat. Tämä ryhmitelmä toistui alkuperäisessäkin suunnitelmassa, mutta koska eri tarvikeryhmien tiedot vaihtelivat huomattavasti keskenään, ei niitä voinut yhdistää yhteen tauluun. Tietokantaa rakennettaessa tulee miettiä, tarvitaanko kaikkia omiin sarakkeisiinsa kirjattuja tietoja. Esimerkiksi tarvikkeiden erilaiset hintatiedot ovat omassa sarakkeessaan, ja hintatietosarakkeita voikin olla useampia yhtä tarviketta kohden. Hintatietoa ei kuitenkaan käytetä varsinaisesti mihinkään muuhun, kuin vain arkistointiin. Hintoja ei ole tarkoitus laskea missään vaiheessa, eikä niistä ole tarvetta muodostaa esimerkiksi hintaseurantakaavioita. Mielestäni olisikin syytä tarkasti pohtia, tarvitseeko hintatietoja ensinnäkään kirjata ylös ja toisekseen tarvitseeko niitä sijoittaa omiin sarakkeisiinsa. Eh-

dotukseni jatkokehitystä varten onkin, että hintatiedot ja muut toissijaiset tiedot sijoitetaan lisätieto-kenttään, josta ne tarvittaessa hakusanoja käyttämällä löytyvät. Näin lomaketta ja tietokantaa saataisiin selkeytettyä huomattavasti.

Tietokantaohjelman valinta kohdistui työssäni Microsoftin Accessiin. Käytössäni oli Access 2010-versio, joka tallentaa tehdyt tietokannat .accdb-päätteellä. Prototyyppiä kehitettäessä valmiiksi sovellukseksi, tulen luultavasti korvaamaan .accdb-päätteisen tietokannan .mdb-päätteisellä. Jälkimmäinen tyyppi on myös Access-tietokanta, mutta se on Accessin vanhempaa tiedostomuotoa. Työn loppuvaiheessa testasin .mdb-tietokantaa lyhyesti Visual Studiossa, eikä sen käytössä tämän testauksen perusteella ilmennyt mitään niistä monista ongelmista, joita .accdb-päätteisen tietokannan kanssa oli. Tiedostomuodon mahdollinen vaihtaminen vaatinee laajempaa testausta, mutta mikäli vanha tiedostomuoto toimii ja tekee myös ohjelmoinnista helpompaa, en näe mitään syytä jatkaa uudemman tiedostomuodon käyttöä prototyyppiä kehitettäessä.

6.2 Sovelluksen käytettävyys

Sovelluksen rakenne, toiminnot ja käytettävyys kärsivät ohjelmointiympäristöön liittyneistä ongelmista. Toimeksiantaja pääsee kuitenkin testaamaan prototyyppiä ja haluamiaan toimintoja. Tämän testauksen jälkeen tuleekin palata vaatimusmäärittelyyn ja tarkentaa ja korjata siinä tehtyjä määrittelyjä. Huomio tulee kiinnittää myös käytettävyyteen, joka mielestäni ontuu jonkin verran prototyypissä.

Prototyypissä korin sisältämien tarvikkeiden yhteensopivuus voidaan tarkistaa erillisellä painikkeella. Vaatimusmäärittelyä tarkennettaessa tuleekin pohtia, onko tämä toiminto hyvä sellaisenaan, vai pitäisikö yhteensopivuuden tarkistaminen yhdistää korin tallennustoimintoon. Yhteensopivuuden tarkistamiseen liittyvä toinen ominaisuus prototyypissä on sen yleisluontoisuus. Toimeksiantaja piirsi minulle kaavion, jonka mukaan yhteensopivuuden tarkistaminen tehdään. Tämä kaavio on esitetty liitteessä 2. Prototyypissä koriin lisättyjen tarvikkeiden yhteensopivuutta tarkastettaessa sovellus ei ota kantaa yksittäisten tarvikkeiden yhteensopivuuteen. Korin sisältö jaetaan linsseihin, runkoihin sekä runkojen osiin. Linhof-runkoihin sopii vain Linhof-tarvikkeet. Mikäli koriin on lisätty Linhof-runko, muttei muita Linhof-tarvikkeita, annetaan tästä huomautus. Sovellus siis tarkastaa vain, onko korissa esimerkiksi Linhof-rungon lisäksi Linhof-linssejä

ja/tai -osia. Yhteensopivuuden tarkastamisen suhteen onkin syytä miettiä, riittääkö toiminnon yleisluontoisuus vai onko sen oltava tarkempi. Mikäli toiminnosta halutaan mahdollisimman tarkka, täytyy yhteensopivuustiedot kirjata tietokantaan. Tämä aiheuttaisi muutoksia myös käyttöliittymään, jonka kautta käyttäjä määrittelee, mitkä tarvikkeet ovat keskenään yhteensopivia. Toisaalta yhteensopivuuden tarkastamistoiminnon tarkentamisella koriin voitaisiin lisätä toiminto, joka ehdottaisi käyttäjälle koriin jo lisättyihin tarvikkeisiin yhteensopivia tarvikkeita.

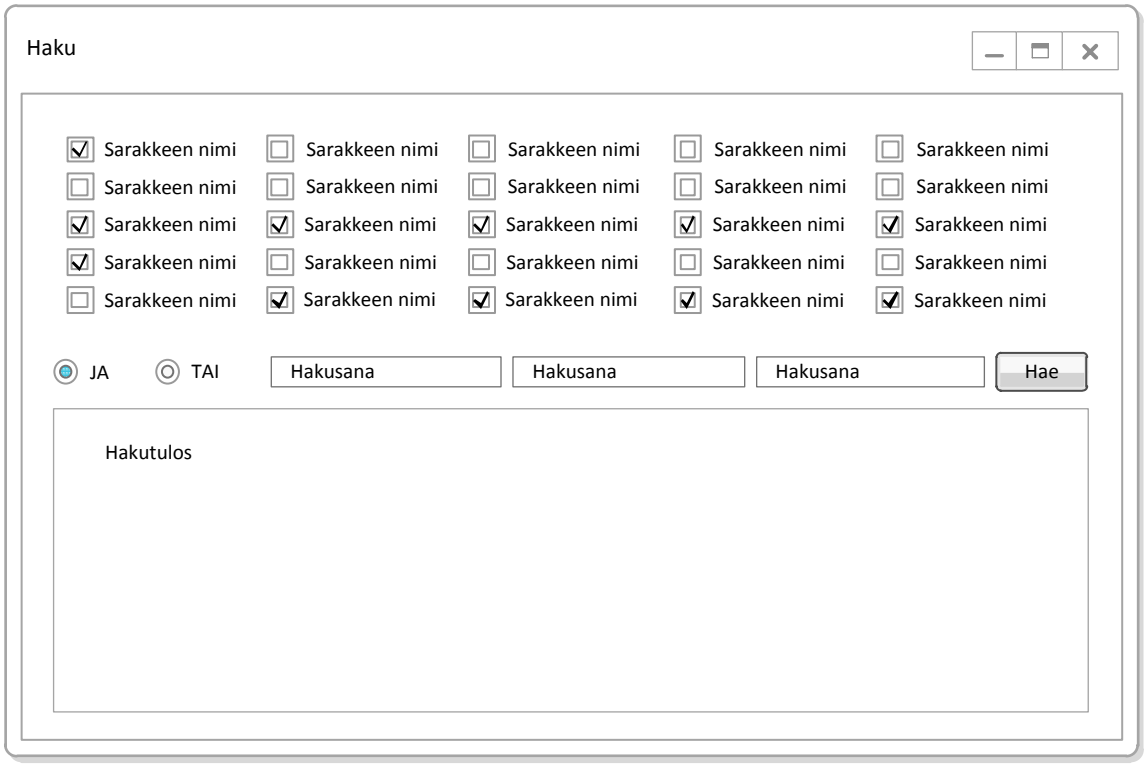
Vaatimusmäärittelyä tarkennettaessa syytä on paneutua myös siihen, mitä tietoja koriin lisätyistä tarvikkeista korissa näytetään. Prototyypissä tarviketiedoista näytetään valmistaja, tarviketyyppi, valmistusnumero, paino, id-numero ja tarvikekategoria. Koska itse en ymmärrä valokuvaustarvikkeiden ominaisuuksista, on toimeksiantajan määriteltävä tarkasti, mitä tietoja hän haluaa korissa näytettävän. Toimeksiantaja onkin jo antanut yhden tarkennuksen tähän liittyen nähtyään prototyypin pikaisesti.

Toimeksiantajan on myös syytä pohtia, tarvitaanko koriin sisällyttää toiminto, jolla saadaan avattua valitun tarvikkeen tarvikekortti. Tämä toiminto olisi hyödyllinen etenkin siinä tapauksessa, että korissa näkyvät tiedot ovat edellä lueteltujen mukaiset. Tarvikekortin avaustoiminto mahdollistaisi tarvikkeen korissa näkymättömien tietojen nopean tarkastamisen.

Prototyypissä korien hallintaikkunan käyttöliittymässä on pyritty siihen, että kaikki toiminnot ovat näkyvillä ja mahdollisimman nopeasti käytettävissä. Mikäli toiminnot olisi sijoitettu esimerkiksi valikkoihin, olisi niiden löytäminen ja käyttäminen vaikeampaa. Toisaalta lomakkeelle sijoitettuina toiminnot tekevät käyttöliittymästä mielestäni hieman vaikeasti hahmotettavan. Helppokäyttöisyyden näkökulmasta järkevintä olisi myös, että korissa käytettäisiin navigaatiopalkkia, joka löytyy kaikkien tarviketietojen hallintaikkunoista. Navigaatiopalkin painikkeet toki liittyisivät korin toimintoihin, mutta esimerkiksi Tallenna-painike voisi olla samanlainen kuin tarviketietojen hallintaikkunoissa. Korien hallintaikkunan käyttöliittymään onkin syytä paneutua tarkemmin jatkokehitysvaiheessa.

Hakutoimintoa ei ole sisällytetty sovelluksen prototyyppiin edellä mainittujen tietokannan ja tietokantayhteyden ongelmien vuoksi. Hakutoiminto tulee kuitenkin sisällyttää varsinaiseen sovellukseen jo senkin takia, että tietyn tarviketiedon etsiminen sadoista

tietueista ilman hakutoimintoa on aikaa vievää. Toimeksiantaja haluaa haun toimivan maksimissaan kolmella sarakeriippumattomalla hakusanalla, hakien kaikista tarviketiedoista. Pikaisesti ajateltuna taulu- ja sarakeriippumaton haku voi vaikuttaa hyvältä ajatukselta, mutta todellisuudessa se voi osoittautua vaikeakäyttöiseksi. ”Sopivia” hakusanoja käyttäen haku voi palauttaa kaikki tietokannan tarviketietueet. Sovellukseen tulisikin mielestäni sisällyttää myös mahdollisuus yksityiskohtaisemman haun muodostamiseen. Yksityiskohtaiseen tai tarkennettuun hakuun käyttäjä voisi esimerkiksi valita sarakkeet, joista tietoa haetaan. Näin käyttäjä voisi hakea tietoa esimerkiksi vain tarvikkeen valmistajan ja valmistusvuoden tai tarviketyypin perusteella. Hakutulosta voisi edelleen tarkentaa mahdollistamalla AND ja OR määritteiden valinta. Kuvassa 9 on esitetty ehdotus hakutoiminnon käyttöliittymälle. Kuvan mukainen käyttöliittymä mahdollistaisi sekä toimeksiantajan toivomusten mukaisen hakutoiminnon että tarkennetun hakutoiminnon. Mikäli lomakkeelle syötetään vain hakusanat, suoritetaan haku niiden perusteella kaikissa tarviketauluissa ja kaikissa sarakkeissa. Lomakkeelta voidaan kuitenkin valita myös sarakkeet, joista tietoa halutaan hakea ja tarvittaessa hakua voidaan tarkentaa valitsemalla AND tai OR määritteet. Hakutulos näytetään lomakkeen alareunassa.



Kuva 9. Hakutoiminnon käyttöliittymäehdotus

Liitteessä 17 on esitetty prototyypin testausta varten tekemäni käyttöohje. Tämä käyttöohje on tehty pääosin tulostusversiota silmälläpitäen. Samaa käyttöohjetta voi kuitenkin lukea ja käyttää myös sovelluksen prototyypin kautta. Jatkokehityksen aikana on syytä pohtia keinoja, miten käyttöohje voitaisiin rakentaa itse sovellukseen paremmin. Yhden tiedoston lukeminen sovelluksen kautta ei ole paras mahdollinen ratkaisu. Käytettävyysasiantuntija Jakob Nielsen on ottanut kantaa käytön ohjeistukseen ja hän neuvoo tekemään dokumenteista sekä näytöltä että paperitulosteelta luettavat versiot (Nielsen 2000, 94). Näytöltä luettavan käyttöohjeen tulee olla yhtä kattava ja opastava kuin paperiversionkin, mutta sen täytyy olla lyhyempi ja tietojen jaotteluun tulee kiinnittää enemmän huomiota.

6.3 Virhetilanteiden ja -toimintojen estäminen

Koska sovelluksen prototyypin pohjalla on tietokanta vain osittain, on erilaisten virhetilanteiden syntyminen mahdollista ja jopa todennäköistä. Esimerkkinä virhetilanteesta on korien hallinta. Koska en saanut tietokantaa toimimaan alkuperäisen suunnitelman mukaisesti, tallentuu korin sisältö tietokannan sijaan tekstitiedostoon. Tämä taas johtaa siihen, että jos korissa olevien tarvikkeiden tietoja muutetaan tietokantaan, ne eivät päivity koriin. Toki olisi mahdollista luoda koriin toiminto, joka tarkastaa tarvikkeiden tiedot tietokannasta esimerkiksi koria avattaessa. Mutta koska ohjelman käytön ja toimivuuden sekä ohjelmoinnin kannalta on järkevämpää tallentaa korien tiedot tietokantaan, ei tekstitiedoston sisältämän tiedon tarkastamiseen kannata kuluttaa aikaa.

Yksi kehityskohde ja keskeinen virhetilanteiden aiheuttaja on sovelluksen sulkeminen. Prototyyppi ei huomauta suljettaessa, mikäli tietoja ei ole tallennettu. Tämä voikin aiheuttaa pahimmillaan tuntien työn häviämisen. Lomakkeiden FormClosing-tapahtumaan tulee ehdottomasti lisätä virhetilanteita estävä toiminto, joka siis suoritetaan kun lomaketta ollaan sulkemassa. Tämän tyyppisiä toimintoja on esimerkiksi tekstinkäsittelyohjelmissa, joissa suljettaessa huomautetaan, mikäli tiedoston tiedot ovat muuttuneet ja annetaan käyttäjälle mahdollisuus joko tallentaa tehty työ tai ohittaa tallennus.

Virheentarkastus tulee jatkokehityksessä ulottaa myös sovellusikkunoiden eri kenttiin. Prototyypin testaamisen jälkeen toimeksiantajan kannattaa miettiä, minkä muotoista tietoa hän eri kenttiin ja lomakkeisiin haluaa syöttää. Prototyypissä tarviketiedot ovat

tekstimuodossa painoa lukuun ottamatta. Sovelluksen on syytä tarkastaa esimerkiksi, että kenttiin on syötetty sallittu määrä merkkejä. Tätä ei ole prototyypissä vielä huomioitu. Samoin osittain huomiotta on jäänyt paino-kentän tarkastus. Prototyyppi huomauttaa, mikäli kenttään lisätään muuta kuin numeromuotoista tietoa. Tekstimuotoinen tieto menee kuitenkin tallennus-toiminnosta läpi, eli vaikka väärästä tietotyypistä huomautetaan, sovellus kuitenkin sen lopulta hyväksyy. Tämän tyyppiset ristiriitaiset ominaisuudet tulee jatkokehityksessä huomioida ja poistaa.

7 TULOSTEN ANALYSOINTI JA POHDINTA

Opinnäytetyön tekeminen oli haasteellisempaa kuin odotin. Tiukka aikataulu aiheutti monia ongelmia ja vaikutti työn laatuun ja tulokseen negatiivisesti. Tavoiteltu tulos on kuitenkin työssä saavutettu. Tässä viimeisessä kappaleessa analysoinkin opinnäytetyön tuloksia erilaisista näkökulmista. Pohdin myös työssä ilmenneitä ongelmia ja työn vaiheita.

7.1 Tulosten analysointi

Ajateltaessa työn tulosta konstruktiivisen tutkimusmenetelmän periaatteiden näkökulmasta, voidaan sanoa, että tutkimus on onnistunut. Suuressa mittakaavassa ajateltuna tuotettu sovelluksen prototyyppi ei sisällä merkittävää uutuusarvoa. Kuitenkin toimeksiantajan näkökulmasta katsottuna prototyypin pohjalta kehitettävällä sovelluksella on käytännön tehtävien suorittamisen kannalta uutuusarvoa. Sovellus auttaa työtehtävien hallinnassa paremmin kuin aiemmin käytössä ollut ohjelmisto. Voidaankin sanoa, että opinnäytteessä tieteellinen tutkimus on onnistunut, koska työssä on kyetty käyttämään jo olemassa olevaa teoriatietoa uudella tavalla.

Jos työn tulosta ajatellaan siihen kohdistuneiden vaatimusten näkökulmasta, on tulos onnistunut. Karkeasti määriteltynä opinnäytetyö sisältää ongelman, ratkaisun kehittämiseen määrätyn ajan, tuotetun ratkaisun ja työn raportoinnin. Opinnäytetyöni sisältää kaikki nämä ominaisuudet. Ongelmana oli toimeksiantajayrityksen laajan tarvikekannan riittävä hallinta. Ratkaisun kehittämiseen aikaa oli vuoden 2010 syyskuun puolesta välistä saman vuoden marraskuun loppuun asti. Määrätyssä ajassa ja määrättyyn ongelmaan kehitin ratkaisuksi työpöytäsovelluksen prototyypin. Työn raportointi mukailee sille määriteltyjä vaatimuksia ja sen tavoitteena on olla sovelluksen prototyypin jatkokehityksen apuna. Myös raportointi on tehty määrätyn ajan sisällä.

Työn tulosta voidaan tarkastella myös sille asetettujen toimintavaatimusten valossa. Myös tällöin työn tulos on onnistunut. Toimeksiantajan vaatimukset liittyivät tarvikekannan hallintaan, työtehtävien suunnitteluun ja valmisteluun ja tarvikkeiden yhteensopivuuden tarkistamiseen. Sovelluksen prototyypissä on toimintoja ja ominaisuuksia, jotka vastaavat näihin päävaatimuksiin.

Jos kuitenkin tarkastellaan työn tulosta pintaa syvemmältä, huomataan sen sisältävän puutteita. Opinnäytetyön aikataulu oli niin tiukka, etten pystynyt keskittymään työhön niin syvällisesti kuin olisin halunnut. Aikataulu ei esimerkiksi antanut mahdollisuutta tutkia sovelluksen rakennetta kovinkaan tarkasti, vaan rakenne jäi sille tasolle, jonka projektin alussa määrittelin. Opinnäytetyön aikataulutin alun perin niin, että olisin aloittanut työn jo kesäkuussa 2010. Aikataulun sotki kuitenkin perusteellisesti syyskuun puolivälissä pakosta tehty aiheenvaihto. Olin asennoitunut siihen, että minulla on puoli vuotta aikaa tehdä opinnäytetyö. Yhtäkkiä kuitenkin huomasin, että jos haluan valmistua ajoissa, aikaa onkin vain kaksi ja puoli kuukautta. Viisitoista opintopistettä, jonka laajuinen opinnäytetyö on, vastaa noin kahta ja puolta kuukautta. Silti olen sitä mieltä, että aika oli liian lyhyt ja se vaikutti negatiivisesti koko työhön.

7.2 Pohdinta

Opinnäytetyön tekemisessä kohtasin valtavan määrän erilaisia ongelmia. Voinkin rehellisesti sanoa, ettei mikään aiempi projekti tai tehtävä ole ollut niin vaikea ja hitaasti etenevä, kuin tämä opinnäytetyö. Ongelmia oli laitteiden, ohjelmistojen, ohjelmoinnin ja tietokannan kanssa. Kun sain yhden ongelman ratkaistua, oli seuraava ongelma jo odottamassa. Laiteongelmia aiheuttivat luonnollisesti tietokoneet. Olin suunnitellut työstäväni opinnäytettä koulusta saadulla kannettavalla, mutta siinä ei riittänytkään muisti kaikille tarvitsemilleni ohjelmille. Siirryin käyttämään omaa kotikonettani, jolloin törmäsin ohjelmisto-ongelmaan. Visual Studio-ohjelmointiympäristöstä on tarjolla myös ilmainen Express-versio. Express-versiolla pystyy rakentamaan yksinkertaisia ja miksei laajahkojakin sovelluksia. Näin siis siinä tapauksessa, että ohjelman saa toimimaan tietokoneellaan. Ongelma ratkesi koulusta saaduilla Visual Studion asennuslevykkeillä. Opinnäytetyöprosessin loppuvaiheessa sain vielä taistella laitteisto-ongelmien kanssa kun näyttöni ja hiireni hajosivat. Kaiken kaikkiaan laitteisto ja ohjelmisto-ongelmien työstämiseen ja ratkaisemiseen käytin 3-4 päivää aikaa. Uskoisin, että opinnäytetyön tekemisen aikana törmäsin suurempaan määrään ongelmia, kuin mitä keskivertoprojektissa ilmenee. Toisaalta tämä oli kuitenkin hyvä opetus, koska ilman näitä laitteisto-ongelmia en varmasti olisi ikinä ottanut niiden mahdollisuutta tarpeeksi vakavasti.

Tietokannan ja ohjelmoinnin suhteen törmäsin jatkuvasti ongelmiin, joiden ratkaisemiseksi jouduin jatkuvasti venyttämään aikatauluani. Osa ongelmista jäi ratkaisematta, koska aika ei yksinkertaisesti olisi riittänyt. Aikataulusta johtuvat ongelmat siirtyivät suoraan prototyyppiin, enkä olekaan täysin tyytyväinen lopputulokseen. Prototyypin kehityksessä on siis käytetty oikoteitä ja unohdettu osittain hyvä koodauskäytäntö ja tietokannan ja käyttöliittymän rakenne. Onneksi kyseessä on kuitenkin prototyyppi, jonka ei tarvitse ollakaan täydellinen. Puutteellinen prototyyppi tarkoittaa sitä, että puutteet on otettava huomioon jatkokehityksessä. Tämä tietenkin tarkoittaa sitä, että jatkokehityksen suhteen työtä on vielä paljon.

Koska aikaa työn valmistumiselle oli vain kaksi ja puoli kuukautta, tarkoitti se käytännössä sitä, etten pystynyt hetkeksikään päästämään työtä mielestäni. Tämä ei millään tavalla parantanut työn tekoa tai sen tulosta, päinvastoin. Työn kannalta olisi ollut tarpeellista saada välillä siirtää työ syrjään ja keskittyä johonkin muuhun. Tämä olisi antanut mahdollisuuden välillä tarkastella tehtyä työtä ikään kuin puhtaalta pöydältä. Yksintyöskentely oli mielestäni myös turhan haastavaa, varsinkin tiukan aikataulun takia. Ongelmat, joihin törmäsin, olivat lopulta hyvin pieniä. Mutta koska minulla ei ollut työparia, jonka kanssa olisin voinut käydä asioita läpi, muodostui pienistä asioista päivien ja viikkojen työtä vaativia ongelmia. Alkuperäinen suunnitelmani oli tehdä täysin käyttövalmis sovellus. Tiukka aikataulu aiheutti sen, että jouduinkin vaihtamaan tavoitteekseni prototyypin valmistamisen.

Yksi työn tekemisen kannalta keskeisistä ongelmista oli myös tarvikkeiden käyttötarkoitus. Kyseessä on siis erilaiset kamera- ja valokuvaustarvikkeet. Kameroihin ja valokuvaukseen liittyvien asioiden suhteen tietämykseni on hyvin yleistä tasoa. Kun aletaan puhua kameratarvikkeiden ominaisuuksista ja erilaisista lisäosista, mennään jo sille tasolle, josta minulla ei ole mitään käsitystä. Tämä vaikeutti sekä tietokannan että koko sovelluksen suunnittelua. Tietokanta ja sovellus piti siis rakentaa sellaisten tietojen ympärille, joita en ymmärtänyt. Muutama päivä meni pelkästään siihen, että yritin jotenkin hahmottaa tarvikeryhmiä ja saada käsityksen erilaisista tarvikkeista ja niiden ominaisuuksista.

Opinnäytetyötä tehdessäni jouduin useassa vaiheessa ja usean eri asian suhteen miettimään vastausta yhteen sovelluskehityksen ja asiakaslähtöisen työskentelyn kannalta tärkeään kysymykseen. Pohdin kumpi on tärkeämpää sovelluksen suunnittelussa ja to-

teutuksessa, säännöt ja suositukset vai asiakkaan toiveet ja vaatimukset? Opinnäytetyön toimeksiantajalla oli hyvin vahva näkemys siitä, millainen sovelluksen tulisi olla. Näkemys ulottui toiminnallisuudesta myös tietokantaan ja käyttöliittymään. Tämä näkemys ei kuitenkaan ollut oppimieni suunnittelu- ja toteutussääntöjen ja -suositusten mukainen. Toimeksiantaja halusi yksinkertaisuutta ja selkeyttä tarviketietojen hallintaan. Tämä tarkoitti esimerkiksi tietokannan suhteen sitä, että kaikki tarviketiedot kirjataan yhteen tauluun, riippumatta siitä, millaisia tietoja tarvikkeisiin liittyy. Tähän liittyen hän halusi käyttöliittymään vain yhden lomakkeen, jolla tiedot tietokantaan lisätään.

Koska sovellusta tulee käyttämään vain toimeksiantaja itse, olisi se voitu toteuttaa täysin hänen vaatimustensa ja toiveidensa mukaisesti. Näin ei missään nimessä voisi toimia, mikäli kyseessä olisi useamman käyttäjän sovellus. Sovelluksen prototyypissä yhdistyy säännöt ja suositukset sekä toimeksiantajan vaatimukset. Prototyypin rakentamisen pohjalla on sekä koulussa opetettuja asioita että laajahkosti lähdemateriaalia. Tuottamani prototyyppi ei kuitenkaan noudata tarkasti toimeksiantajani näkemystä ratkaisusta, eikä se vastaa täsmällisesti hänen vaatimuksiinsa.

Luvussa 2 käsittelin käytettävyyden näkökulmaa. Sekä Suomen laki että Suomen standardisoimisliitto määrittävät sovelluksen käytettävyyttä ja sovelluksen käyttäjän tai valitsijan vastuuta. ISO-standardit rohkaisevat tunnistamaan lopputuotteen käyttäjäryhmän, käyttäjäsuoritukseen vaikuttavat tekijät sekä käyttötilanteen. Kun nämä on havaittu ja määritelty, voidaan tuote sopeuttaa mahdollisimman tarkoin tarkoitettuun käyttöön sopivaksi. Tähän liittyen Suomen standardisoimisliitto huomauttaa tietotyön ergonomiaa käsittelevässä käsikirjassaan, että ”tuotteen käyttäjällä, hankkijalla tai valitsijalla on lopullinen vastuu siitä, että valitsee ja osaa valita tarvitsemansa tuotteen oikein” (Suomen standardisoimisliitto 2000, 14). Tämän voidaan mielestäni katsoa osittain vapauttavan sovelluskehittäjän vastuusta. Tämä koskee kuitenkin vain tilanteita, joissa asiakkaalle on tarjottu määriteltyyn ongelmaan vastaavia ja toimivia ratkaisuja, joita asiakas ei ole hyväksynyt.

Vaikka edellä mainittujen tahojen mukaan lopullinen vastuu sovelluksen valinnasta on valitsijalla, mietin kuitenkin, kuinka kattava on minun vastuuni sovelluksen kehittäjänä. Jos toimeksiantaja ei hyväksy tuottamaani ratkaisua, mietinkin, onko minun tehtäväni kuitenkin muuttaa hänen mielensä ja ”myydä” tuottamani ratkaisu hänelle. Opintojen aikana meille opetetaan, että tekemämme työt ovat käyntikortti osaamisestamme ja ky-

vyistämme, ja siksi ei pidä myöntyä kaikkiin asiakkaan vaatimuksiin ja toiveisiin. Kuitenkin mielestäni perimmäinen tarkoitus on tuottaa ratkaisuja, joita asiakas haluaa. Onhan asiakkaallakin vastuunsa sen suhteen, millaisia ratkaisuja hän haluaa ja hyväksyy.

Opinnäytetyöprosessin aikana opin uusia asioita muun muassa Visual Basic-ohjelmointikielestä ja ohjelmistokehityksestä. Työn loppuvaiheessa tajusin tehneeni huonoja valintoja ohjelmointiympäristön suhteen. Prototyypin valmistuksessa luotin liikaa Visual Studion graafiseen käyttöliittymään ja sen mahdollistamiin toimintoihin. Kun sovellusta rakennetaan käyttäen käyttöliittymän sisältämiä toimintoja, luo Visual Studio osan koodista automaattisesti. Jos ja kun ohjelmoidut toiminnot eivät toimikaan oikein, on niiden korjaaminen vähintäänkin haasteellista. Automaattinen koodi siis vaikeuttaa ohjelmoitujen ominaisuuksien syvällistä ymmärtämistä, koska kaikki koodi ei ole itse kirjoitettua. Jatkokehityksen aikana kannattaakin jättää Visual Studion graafisen käyttöliittymän hyödyntäminen vain kontrollien lisäämiseen lomakkeelle ja kirjoittaa kaikki muu koodi itse. Vaihtoehtoisesti jatkokehitysvaiheessa voidaan pohtia, olisiko Visual Basic -kieli syytä korvata jollain toisella ohjelmointikielellä.

LÄHTEET

Painetut

Agarwal, Vidya Vrat & Huddleston, James 2008. Beginning VB 2008. From novice to professional. Apress, Berkeley, USA.

Fowler, Martin & Scott, Kendal 2002. UML. Docendo Finland Oy, Jyväskylä.

Haikala, Ilkka & Märijärvi, Jukka 2004. Ohjelmistotuotanto. 10. painos. Talentum Media Oy, Helsinki.

Hirsjärvi, Sirkka & Remes, Pirkko & Sajavaara, Paula 2009. Tutki ja kirjoita. 15. uudistettu painos. Kustannusosakeyhtiö Tammi, Helsinki.

Hovi, Ari & Huotari, Jouni & Lahdenmäki, Tapio 2003. Tietokantojen suunnittelu & indeksointi. Docendo Finland Oy, Jyväskylä.

Hyppönen, Annikki & Ojala, Alice & Joutsi, Jaakko 2007. Tietokoneen käyttötaito 1. Office 2007-ohjelmille. WSOYpro/Docendo, Jyväskylä.

Järvinen, Jani 2008. Visual Studio 2008 -käsikirja. WSOYpro/Docendo, Jyväskylä.

Krug, Steve 2006. Don’t Make Me Think! A Common Sense Approach to Web Usability. 2. painos. New Riders, Berkeley, USA.

McConnell, Steve 1998. Ohjelmistoprojektit – Selviytymisopas. Suomen Atk-kustannus Oy, Espoo.

Nielsen, Jakob 2000. WWW suunnittelu. Oy Edita Ab, Helsinki.

Nielsen, Jakob & Loranger, Hoa 2006. Prioritizing Web Usability. New Riders, Berkeley, USA.

Suomen standardisoimisliitto, 2000. SFS-käsikirja 72. Tietotyön ergonomia. Yleisperiaatteet, kalusteet ja työasema, ohjelmistot, laitteet. 2. painos. Suomen standardisoimisliitto, Helsinki.

Vilkka, Hanna 2005. Tutki ja kehitä. Tammi, Helsinki.

Painamattomat

Avoin lähdekoodi 2010. Luettu 27.9.2010. <http://www.coss.fi/abc/avoin-lahdekoodi>

Frost, Roger 2007. Overview of ISO 9001 and ISO 14001. Luettu 10.10.2010. <http://www.sfs.fi/files/900114001overview.ppt>

FunctionX, Inc. 2010. The Visual Basic Language. Luettu 21.9.2010. <http://www.functionx.com/visualbasic/index.htm>

Graphical user interface 2010. Luettu 4.10.2010. <http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html>

ISO – International Organization for Standardization 2010. About ISO. Luettu 10.10.2010. <http://www.iso.org/iso/about.htm>

Kankaanpää, Timo 2010. UML1 perusteet. Luettu 27.9.2010. <http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston_maarittely/UML1.html>

Konstruktiivinen tutkimusote 2006. Luettu 24.8.2010. <http://lille.haaga-helia.fi/ampedatk/menetelmapaletti/konstruktiivinen.html>

Kyselyiden optimointi – Osa 2: Tietokannan rakenteen suunnittelu 2007. Luettu 9.11.2010. <http://itpro.fi/asiantuntijaryhmat/tietokanta/Lists/Posts/Post.aspx?ID=24>

Laakso, Sari A. & Laakso, Karri-Pekka 2004. Hyvän käyttöliittymän varmistaminen GUIDe-prosessimallilla. Luettu 15.9.2010. <http://www.cs.helsinki.fi/u/salaakso/papers/GUIDe-suomeksi.pdf>.

Meriläinen Juha 2008. Oliopohjainen suunnittelu. Luennot. Kemi-Tornion ammattikorkeakoulu. Liiketalouden ja tietojenkäsittelyn koulutusyksikkö.

OpenOffice.org-toimisto-ohjelmisto 2010. Luettu 7.10.2010. <http://fi.openoffice.org/tuote.html>

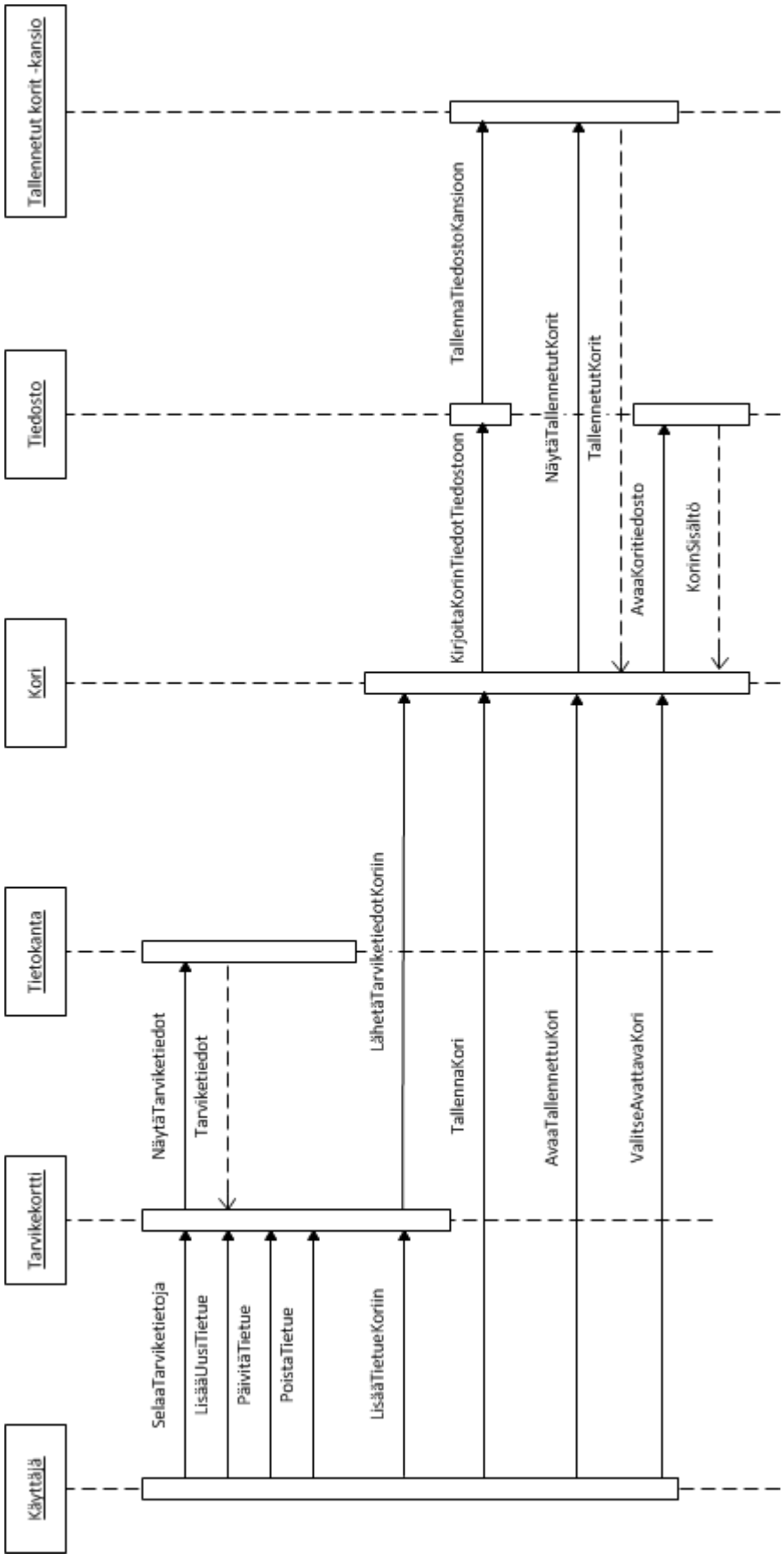
Seppänen, Veikko 2004. Konstruktiivinen tutkimus. Luettu 23.9.2010. <http://media.tol.oulu.fi/video/jtmk/konstruktiivinen_tutkimus.ppt>

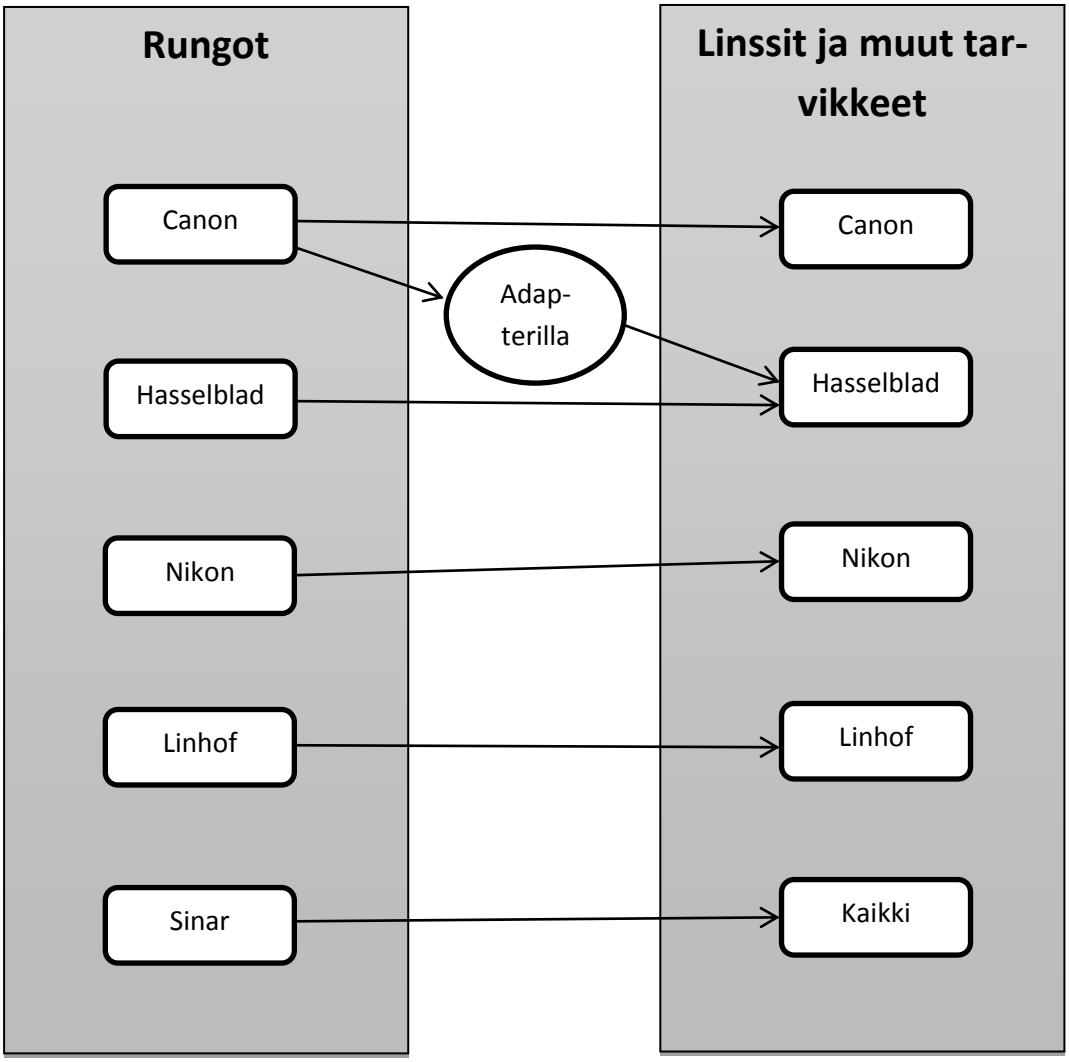
Taina, Juha 2000. Ohjelmistotuotannon prosessimallit. Luettu 14.11.2010. <http://www.cs.helsinki.fi/u/taina/ohtu/k-2001/luennot/prosessi/kaikki.html>

User interface 2010. Luettu 4.10.2010. <http://www.webopedia.com/TERM/U/user_interface.html>

Valtioneuvoston päätös näyttöpäätetyöstä 1983. 22.12.1993/1405. Luettu 10.10.2010. <http://www.finlex.fi/fi/laki/ajantasa/1993/19931405>

LIITTEET

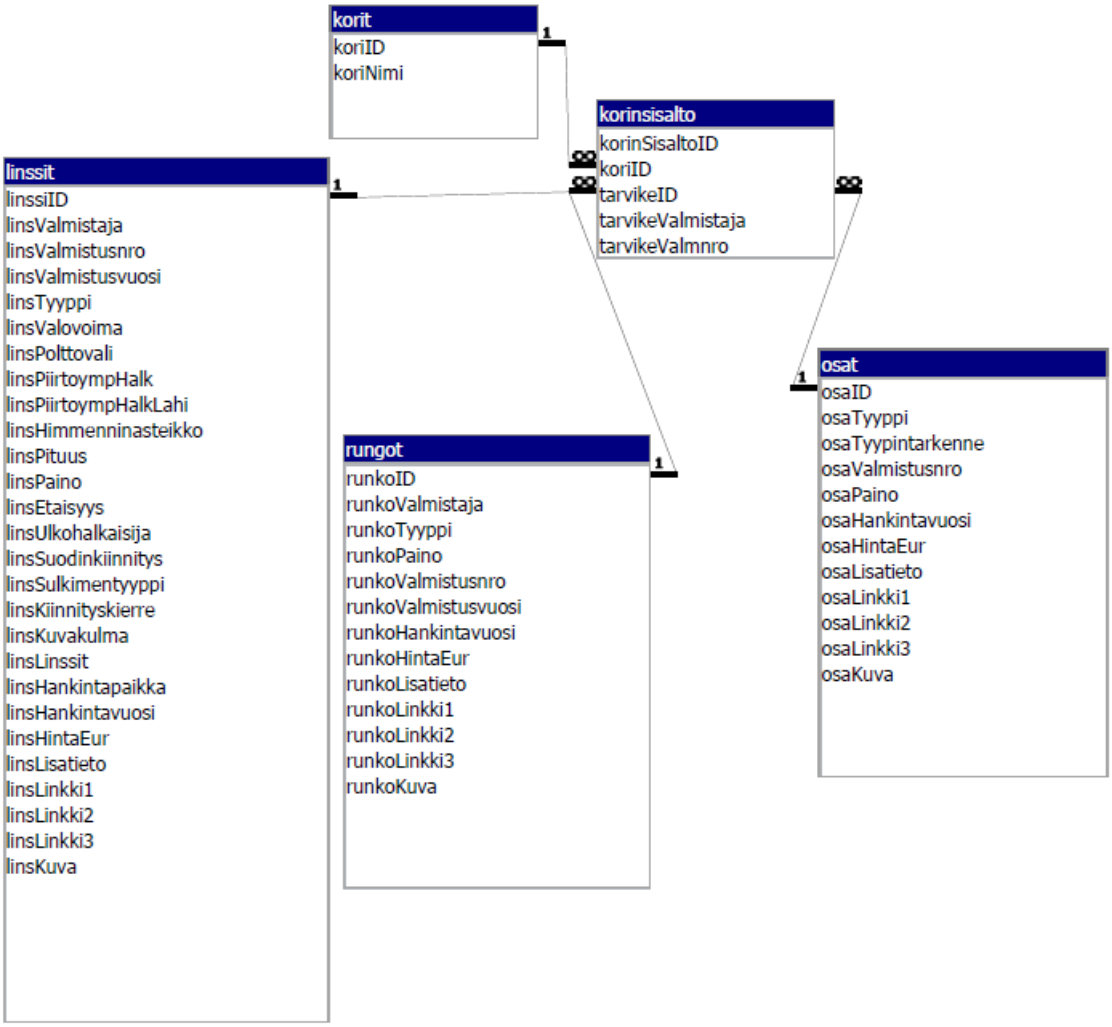

Tallennetut korit -kansio
Tiedosto
Kori
Tietokanta
Tarvikekortti
Käyttäjä
TallennaTiedostoKansioon
NäytäTallennetutKorit
TallennetutKorit
AvaaKoritiedosto
KorinSisältö
KirjoitaKorinTiedotTiedostoon
LähetäTarviketiedotKoriin
NäytäTarviketiedot
Tarviketiedot
TallennaKori
AvaaTallennettuKori
ValitseAvattavaKori
SelaaTarviketietoja
LisääUusiTietue
PäivitäTietue
PoistaTietue
LisääTietueKoriin

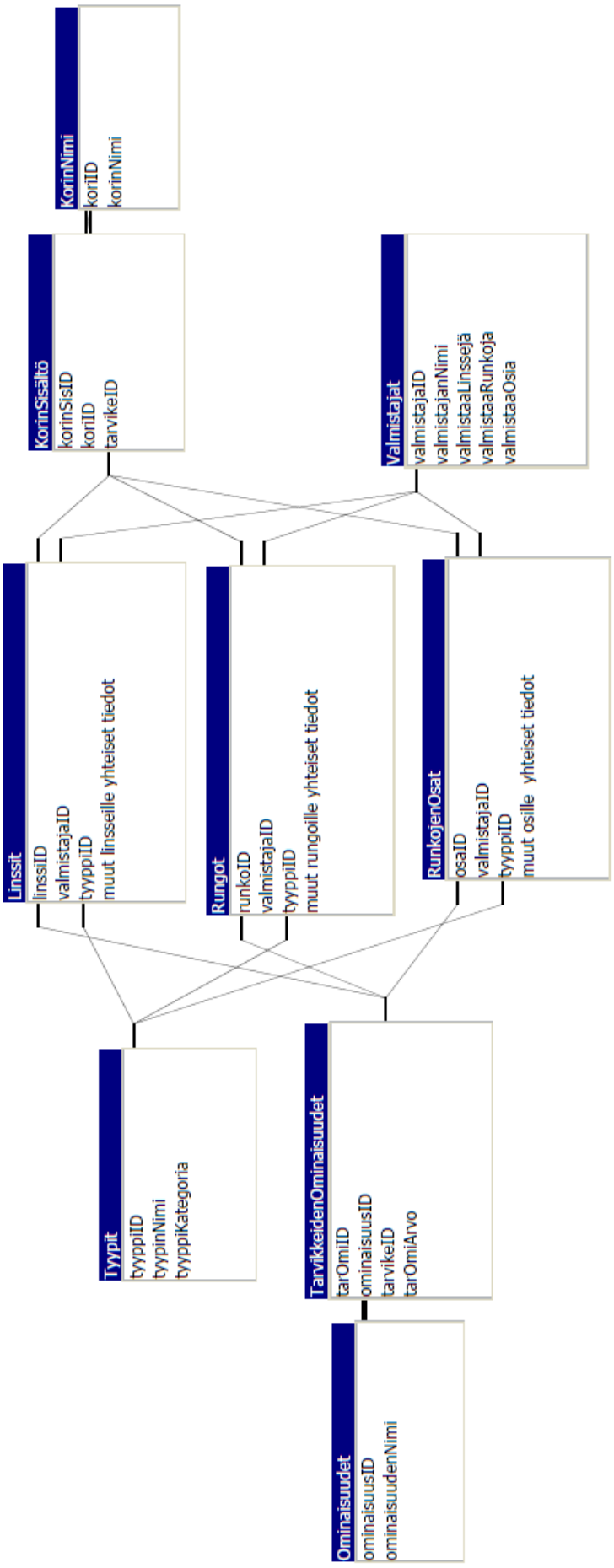

Rungot
Linssit ja muut tar-vikkeet
Canon
Hasselblad
Nikon
Linhof
Sinar
Adap-terilla
Canon
Hasselblad
Nikon
Linhof
Kaikki

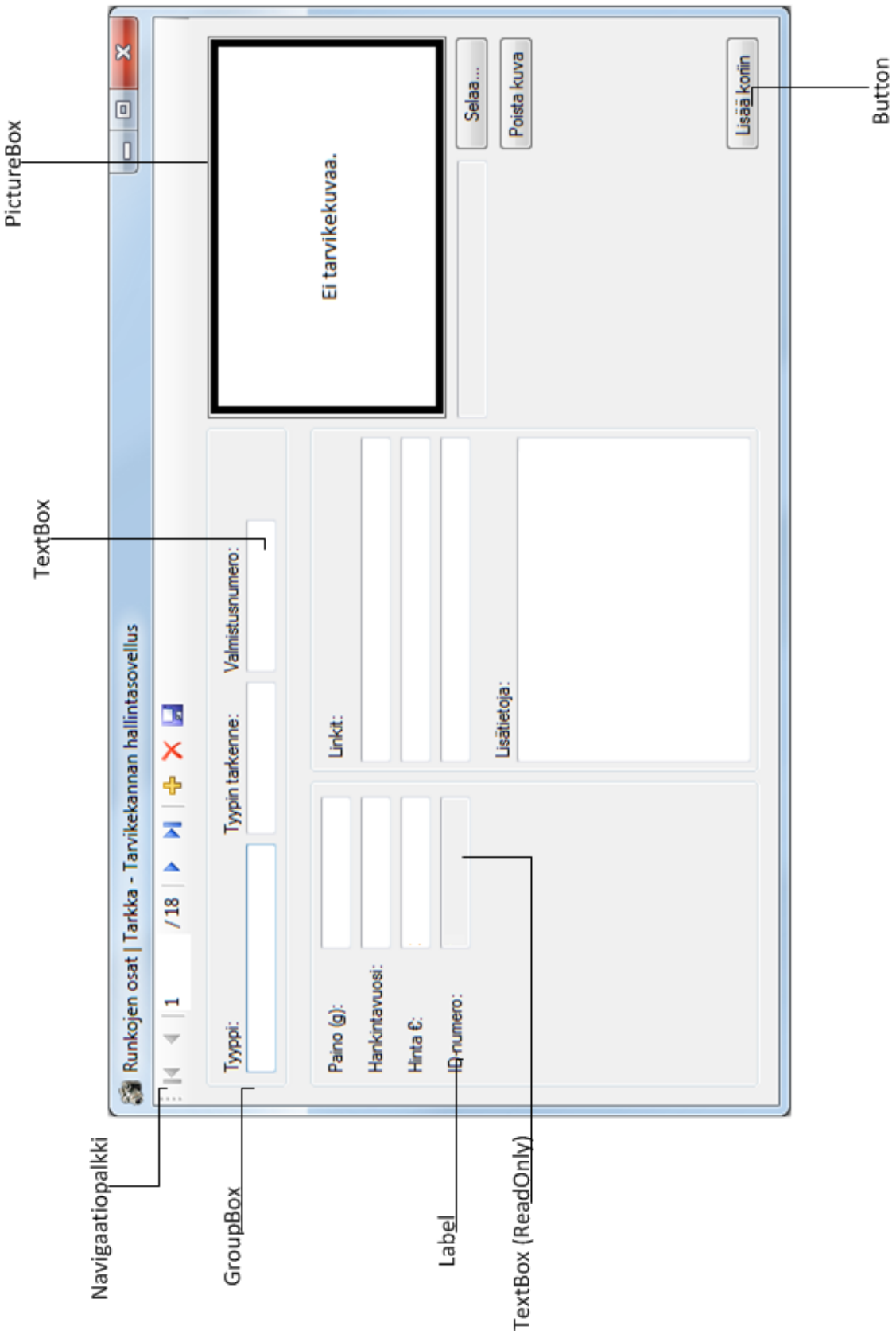
Pääkategoriat	Alakategoriat
LF	- Rungot - Linssit - Runkojen osat
MF	- Hasselblad - Asahi Pentax - Rollerflex
EOS	
FD	- Canon FD runkoja - Canon FD linssit/objektiivit - Canon EX - Canon lisätarvikkeet
Muut 24x36	- EXA - Nikon - Contax - Leica - Pokkarit
OPTIK	
EL	- Linssit - Suurennuskoneet /rungot - Irtolinssit
B+S	- Salamalaitteet - Irtovalotusmittarit - Jalustat - Kinopäät & kuulapolvet - Laukut

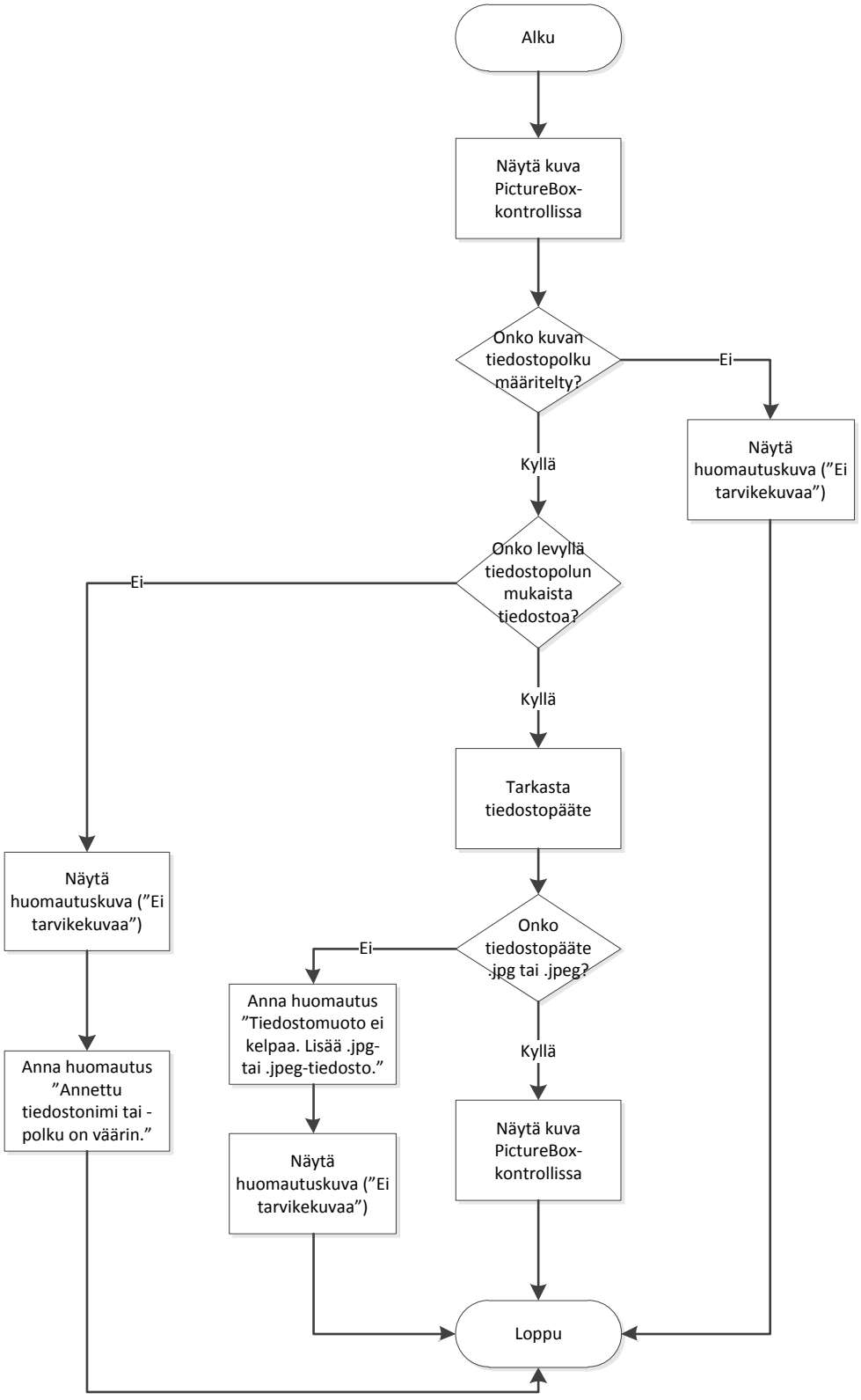
Tarviketiedot	**Large Format-linssit**	**CanonFD -linssit**
Adoraman (kauppa) hinta		x
Etäisyys	x	
Hankintapaikka	x	x
Hankintavuosi	x	x
Himmenninasteikko	x	
Himmentimen lehtien lukumäärä		x
Hinta €	x	x
Hinta jeneinä		x
Kiinnityskierre	x	
Kuva (tarvikkeesta)	x	x
Kuvakulma	x	
Linkki 1	x	x
Linkki 2	x	x
Linkki 3	x	x
Linssiryhmät		x
Linssit	x	
Lisätietoja	x	x
Paino	x	x
Piirtoympyrän halkaisija	x	
Piirtoympyrän halkaisija lähietäi-syydeltä	x	
Pituus	x	
Polttoväli	x	x
Sulkimen tyyppi	x	
Suodinkiinnitys	x	
Tyypin tarkennus		x
Tyyppi	x	x
Ulkohalkaisija	x	
Valmistaja	x	
Valmistettu vuodesta		x
Valmistusnumero	x	x
Valmistusvuosi	x	
Valovoima	x	x

Sovelluksen prototyypissä on käytetty vain Large Format-ryhmän tarviketietoja.

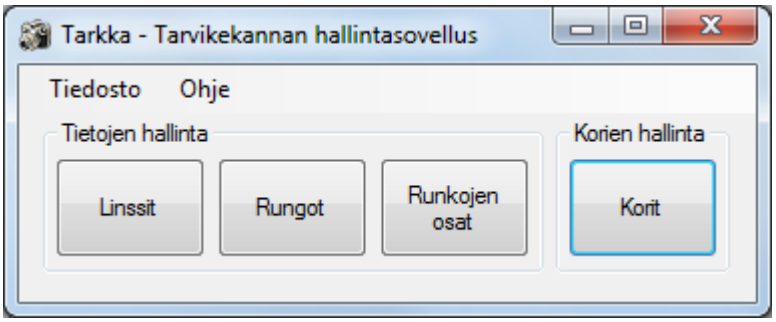

korit
koriID
koriNimi
korinsisalto
korinSisaltoID
koriID
tarvikeID
tarvikeValmistaja
tarvikeValmnro
linssit
linssiID
linsValmistaja
linsValmistusnro
linsValmistusvuosi
linsTyyppi
linsValovoima
linsPolttovali
linsPiirtoympHalk
linsPiirtoympHalkLahi
linsHimmenninasteikko
linsPituus
linsPaino
linsEtaisyys
linsUlkohalkaisija
linsSuodinkiinnitys
linsSulkimentyyppi
linsKiinnityskierre
linsKuvakulma
linsLinssit
linsHankintapaikka
linsHankintavuosi
linsHintaEur
linsLisatieto
linsLinkki1
linsLinkki2
linsLinkki3
linsKuva
rungot
runkoID
runkoValmistaja
runkoTyyppi
runkoPaino
runkoValmistusnro
runkoValmistusvuosi
runkoHankintavuosi
runkoHintaEur
runkoLisatieto
runkoLinkki1
runkoLinkki2
runkoLinkki3
runkoKuva
osat
osaID
osaTyyppi
osaTyypintarkenne
osaValmistusnro
osaPaino
osaHankintavuosi
osaHintaEur
osaLisatieto
osaLinkki1
osaLinkki2
osaLinkki3
osaKuva
1


KorinNimi
koriID
korinNimi
KorinSisältö
korinSisID
koriID
tarvikeID
Valmistajat
valmistajaID
valmistajanNimi
valmistaaLinsseja
valmistaaRunkoja
valmistaaOsia
Linssit
linssiID
valmistajaID
tyyppiID
muut linsseille yhteiset tiedot
Rungot
runkoID
valmistajaID
tyyppiID
muut rungoille yhteiset tiedot
RunkojenOsat
osaID
valmistajaID
tyyppiID
muut osille yhteiset tiedot
Tyypit
tyyppiID
tyypinNimi
tyyppiKategoria
TarvikkeidenOminaisuudet
tarOmiID
ominaisuusID
tarvikeID
tarOmiArvo
Ominaisuudet
ominaisuusID
ominaisuudenNimi

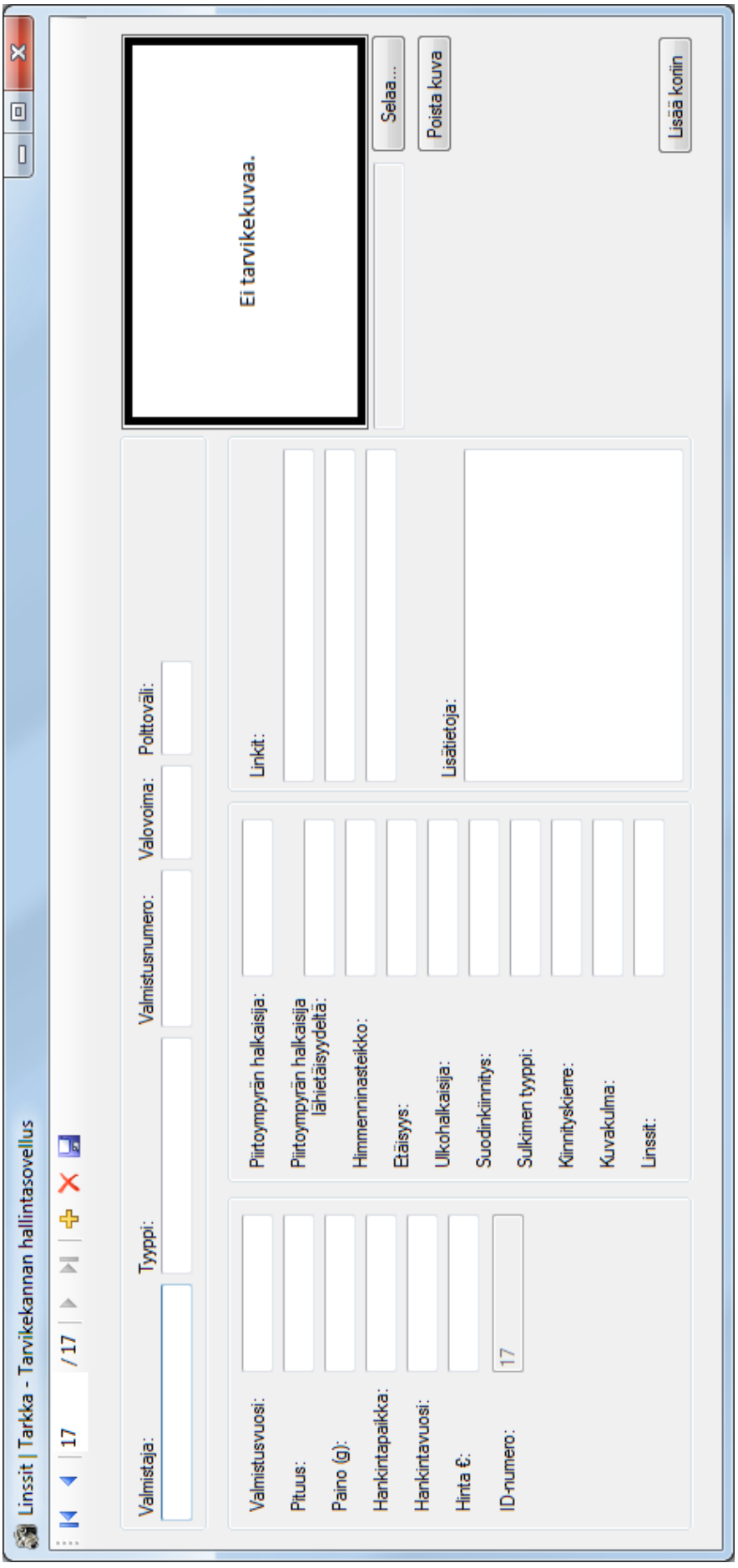

PictureBox
TextBox
Navigaatiopalkki
GroupBox
Label
TextBox (ReadOnly)
Button
Runkojen osat | Tarkka - Tarvikekannan hallintasovellus
Ei tarvikekuvaa.
Selaa ...
Poista kuva
Lisää koriin
Valmistusnumero:
Tyypin tarkenne:
Tyyppi:
Linkit:
Lisätietoja:
Paino (g):
Hankintavuosi:
Hinta €:
ID-numero:

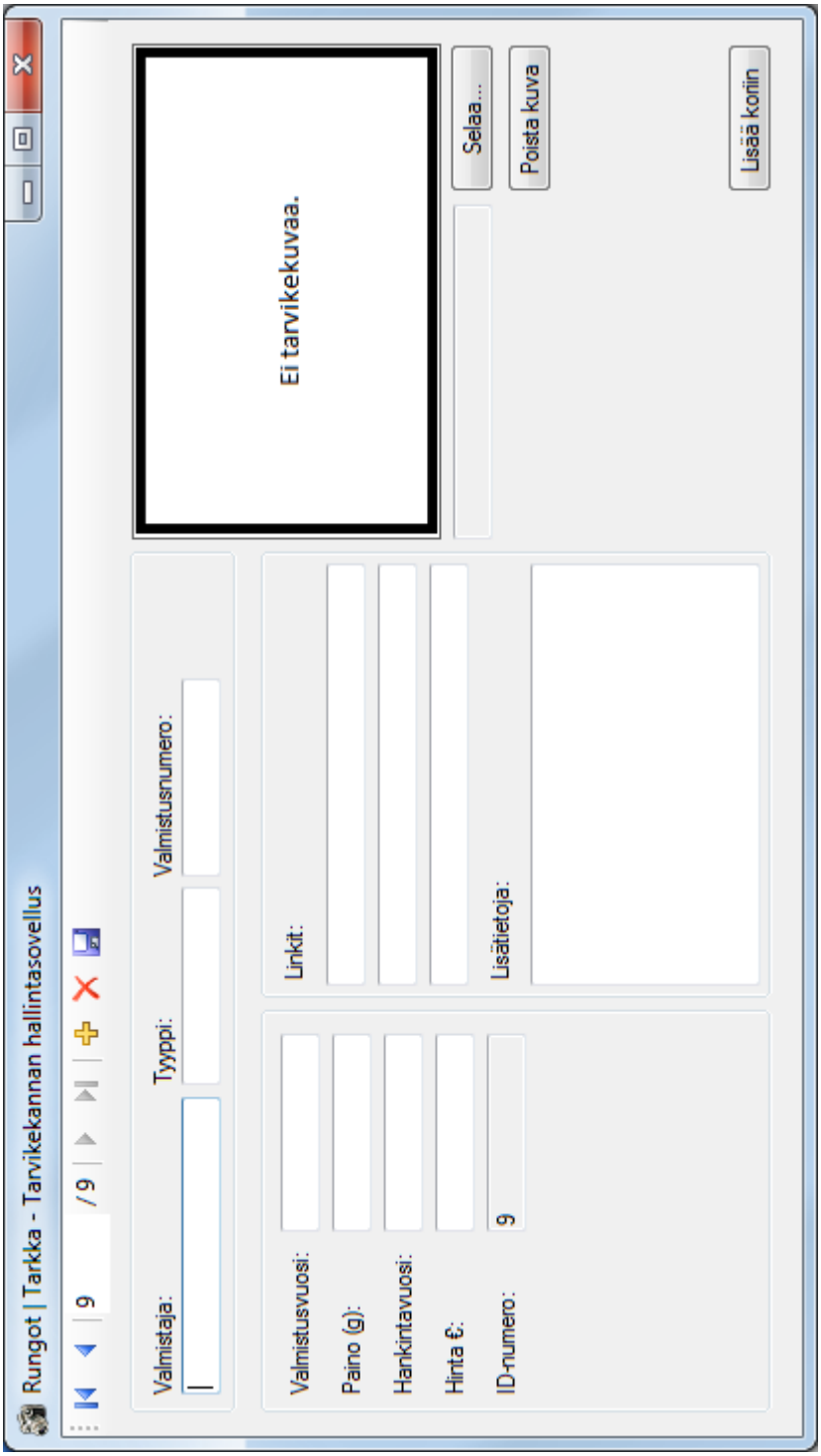

Alku
Näytä kuva PictureBox-kontrollissa
Onko kuvan tiedostopolku määritelty?
Ei
Näytä huomautuskuva (”Ei tarvikekuvaa”)
Kyllä
Onko levyllä tiedostopolun mukaista tiedostoa?
Ei
Kyllä
Tarkasta tiedostopääte
Näytä huomautuskuva (”Ei tarvikekuvaa”)
Onko tiedostopääte .jpg tai .jpeg?
Ei
Anna huomautus ”Tiedostomuoto ei kelpaa. Lisää .jpg- tai .jpeg-tiedosto.”
Kyllä
Anna huomautus ”Annettu tiedostonimi tai -polku on väärin.”
Näytä kuva PictureBox-kontrollissa
Näytä huomautuskuva (”Ei tarvikekuvaa”)
Loppu

```
' Aliohjelma kuvan näyttämiseksi PictureBox-kontrollissa
Private Sub ShowPictureInBox()
     If OsaKuvaTextBox.Text = "" Then
          PictureBoxOsat.Image = My.Resources.eikuvaa
     Else
          If My.Computer.FileSystem.FileExists(OsaKuvaTextBox.Text) = True
          Then
               Dim TiedostoPaate As String =
               System.IO.Path.GetExtension(OsaKuvaTextBox.Text)

               If TiedostoPaate.ToLower = ".jpg" Or TiedostoPaate.ToLower =
               ".jpeg" Then
                        PictureBoxOsat.Image =
                        Image.FromFile(OsaKuvaTextBox.Text)
               Else
                        PictureBoxOsat.Image = My.Resources.eikuvaa
                        OsaKuvaTextBox.Clear()
                        MsgBox("Tiedostomuoto ei kelpaa. Lisää .jpg- tai
                        .jpeg-päätteinen tiedosto.", MsgBoxStyle.Critical)
               End If

          Else
               PictureBoxOsat.Image = My.Resources.eikuvaa
               MsgBox("Kuvan näyttäminen epäonnistui. Annettu tiedostonimi
               tai -polku on väärin.", MsgBoxStyle.Critical)
          End If

     End If
End Sub


' Tietojen tallennus
Private Sub OsatBindingNavigatorSaveItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles OsatBindingNavigatorSaveItem.Click

     Try
          Me.Validate()
          Me.OsatBindingSource.EndEdit()
          Me.TableAdapterManager.UpdateAll(Me.OsatDataSet)
          MsgBox("Tietojen tallennus onnistui!", MsgBoxStyle.Information)
     Catch ex As Exception
          MsgBox("Tietojen tallennus ei onnistunut. " & ex.Message,
          MsgBoxStyle.Critical)
     End Try

End Sub
```

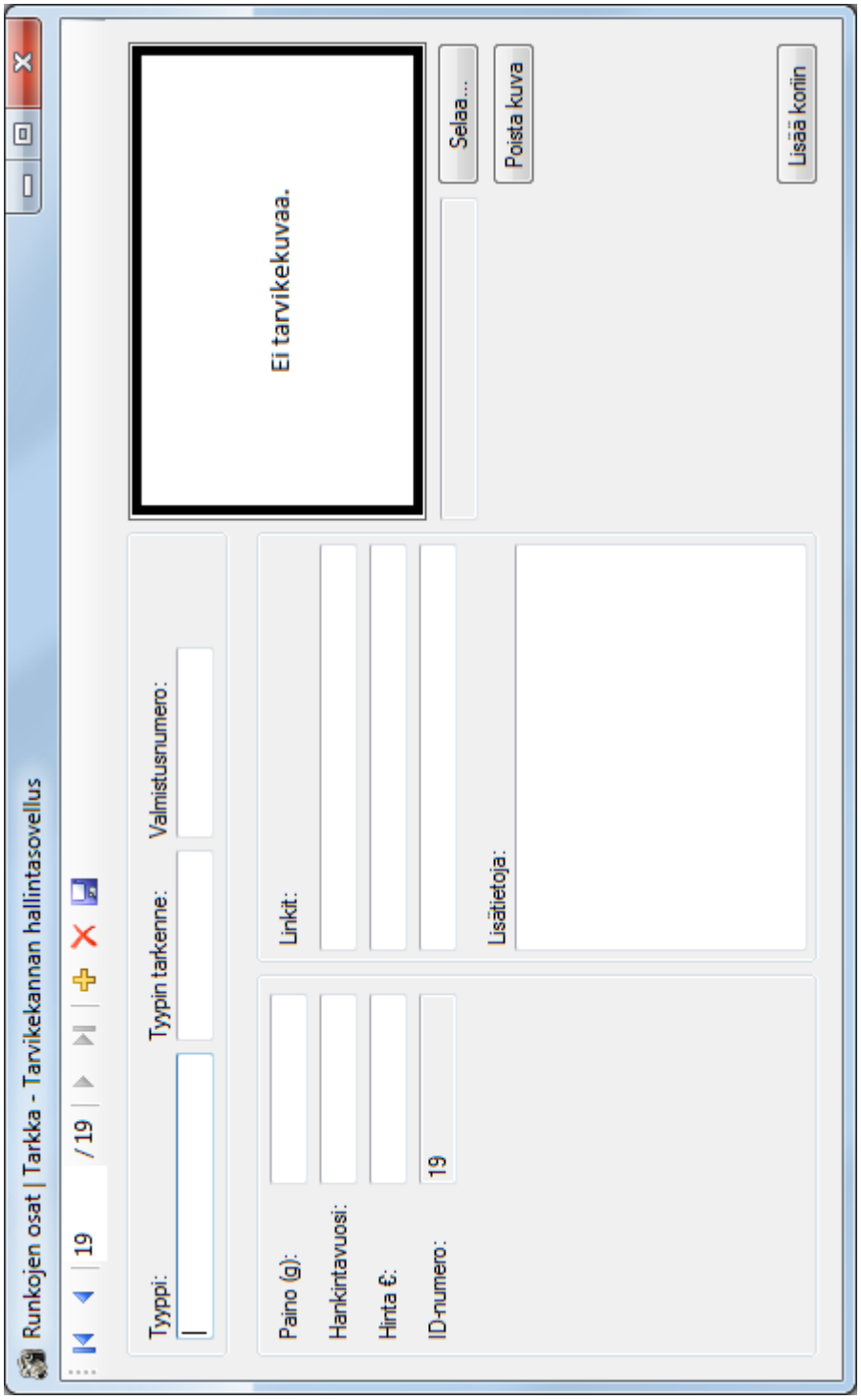

Tarkka - Tarvikekannan hallintasovellus
Tiedosto
Ohje
Tietojen hallinta
Linssit
Rungot
Runkojen osat
Korien hallinta
Korit

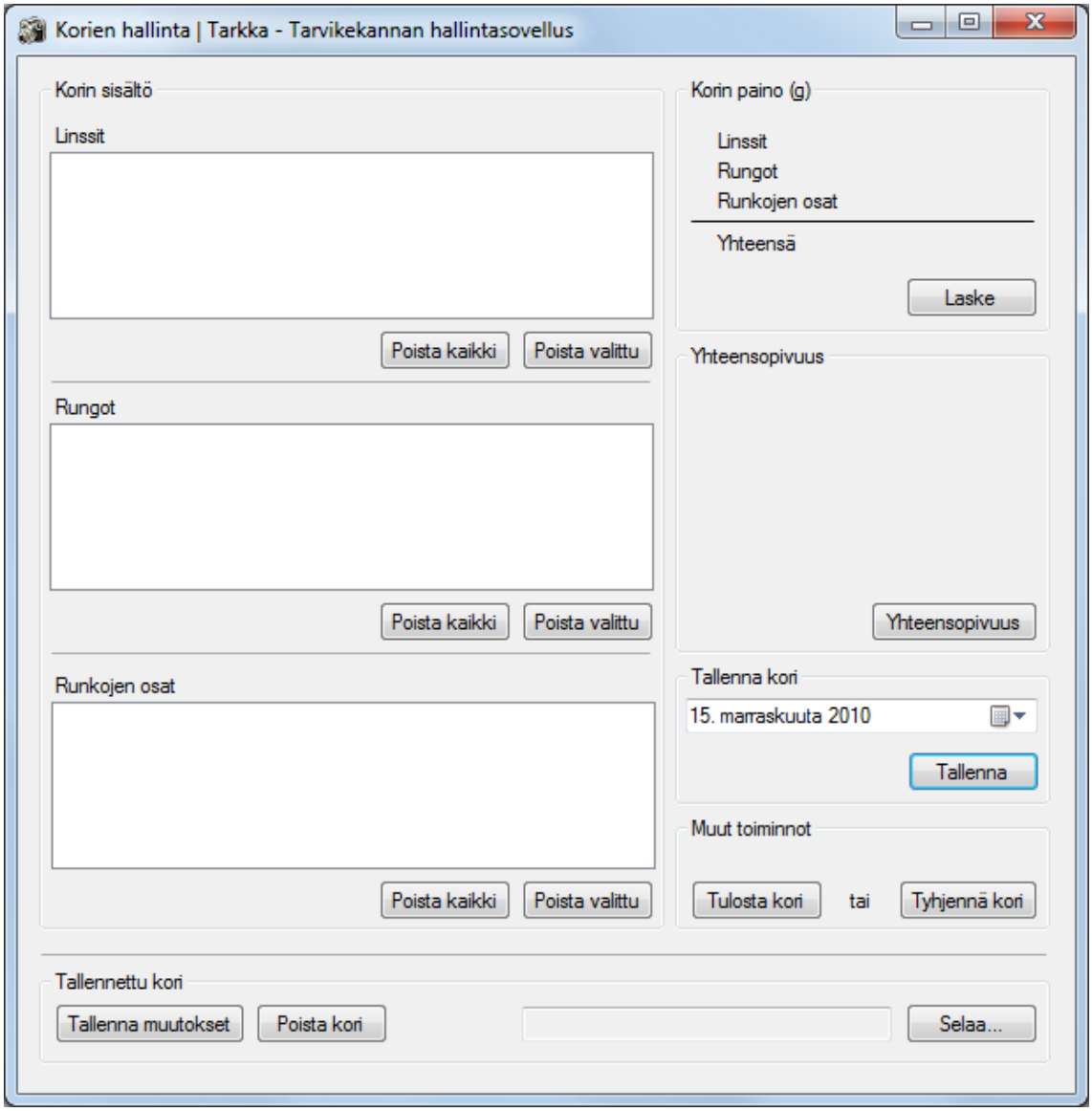
Linssien hallintaikkuna




Rungot | Tarkka - Tarvikekannan hallintasovellus
9
/ 9
Valmistaja:
Tyyppi:
Valmistusnumero:
Ei tarvikekuvaa.
Selaa...
Poista kuva
Linkit:
Lisätietoja:
Lisää koriin
Valmistusvuosi:
Paino (g):
Hankintavuosi:
Hinta €:
ID-numero:
9

Runkojen osien hallintaikkuna




Korien hallinta | Tarkka - Tarvikekannan hallintasovellus
Korin sisältö
Linssit
Poista kaikki
Poista valittu
Rungot
Poista kaikki
Poista valittu
Runkojen osat
Poista kaikki
Poista valittu
Korin paino (g)
Linssit
Rungot
Runkojen osat
Yhteensä
Laske
Yhteensopivuus
Yhteensopivuus
Tallenna kori
15. marraskuuta 2010
Tallenna
Muut toiminnot
Tulosta kori
tai
Tyhjennä kori
Tallennettu kori
Tallenna muutokset
Poista kori
Selaa...

Tarvikekannan hallintasovellus **Test-case 1: .jpg-kuvan lisääminen tietokantaan/tarvikekorttiin**	
Purpose / Tarkoitus:	.jpg-kuvan lisääminen tietokantaan ja tarvikekorttiin
Prereq / Alkutilanne:	Käyttäjällä on auki tarvikekortti (tarvikkeiden hallintaikkuna), joko uusi tarviketieto tai jo olemassa olevan päivittäminen
Test data / Testiaineisto:	Tarvikekortti (tarvikkeiden hallintaikkuna) jpg- ja/tai jpeg-kuvatiedosto Muita tiedostoja (esimerkiksi .png, .exe, .txt, .avi…)
Steps / Vaiheet:	1. Klikataan Selaa… -painiketta 2. Valitaan oikea kohdekansio 3. Valitaan lisättävä tiedosto ja klikataan Avaa -painiketta 4. Tallennetaan tehdyt muutokset Tallenna -painiketta klikkaamalla 5. Selataan tarvikkeita eteen- ja taaksepäin 6. Palataan tarvikkeeseen, johon kuva lisättiin
Expected results / Odotetut tulokset:	Sovellus ei anna lisätä muita kuin .jpg-/.jpeg-kuvatiedostoja tarvikekorttiin. Jos yritetään lisätä muunlainen tiedosto, sovellus antaa huomautuksen eikä lisää tiedostoa tarvikekortille. Sovelluksen hyväksyessä lisätyn tiedoston, kuva näytetään PictureBox-kontrollissa ja kuvatiedoston tiedostopolku PictureBoxin alapuolella olevassa TextBox-kontrollissa. Tallennettu kuva näkyy edellä olevan kuvauksen mukaisesti tietoja selattaessa.
Test results (date/result tester) / Testitulokset (pvm/tulos/testaaja):	

KÄYTTÖOHJE

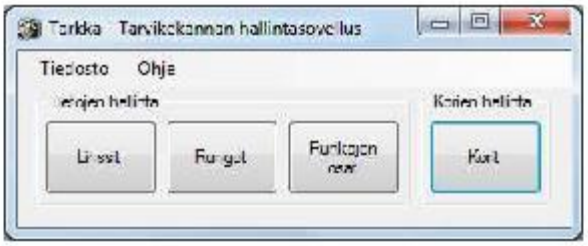
TarkkaProto 1.0

11 / 2010

Kirjoittanut Jenni Lounela

Sisällysluettelo

1 Sovelluksen käytön aloitus

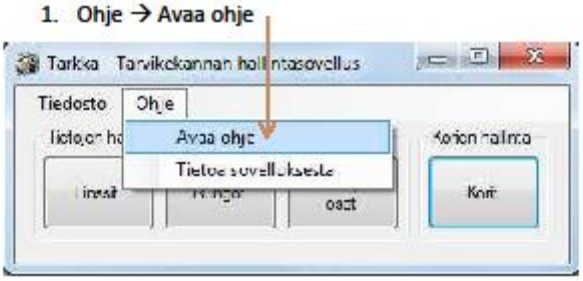


Sovelluksen aloitusruutu sisältää neljä painiketta: Linssit, rungot, runkojen osat ja korit. Tietojen hallinta -ryhmän painikkeista pääsee tarviketietojen hallintaikkunoihin eli tarvikekorteille. Korit-painikkeesta avautuu Korien hallinta -ikkuna.

1.1 Käyttöohje

Käyttöohjeen saa näkyviin aloitusruudun valikkorivin kautta:

1. Ohje → Avaa ohje



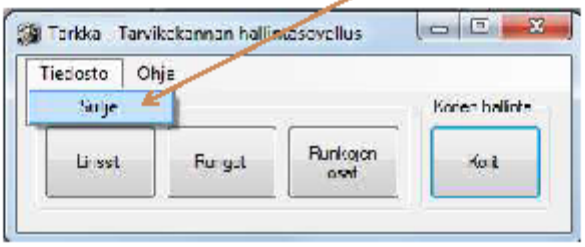
1.2 Sovelluksen sulkeminen

Sovelluksen voi sulkea kahdella eri tavalla:

1. **Klikkaamalla aloitusruudun vasemmassa ylänurkassa olevaa ruksia**

   tai

2. **Valikkorivin kautta: Tiedosto → Sulje.**

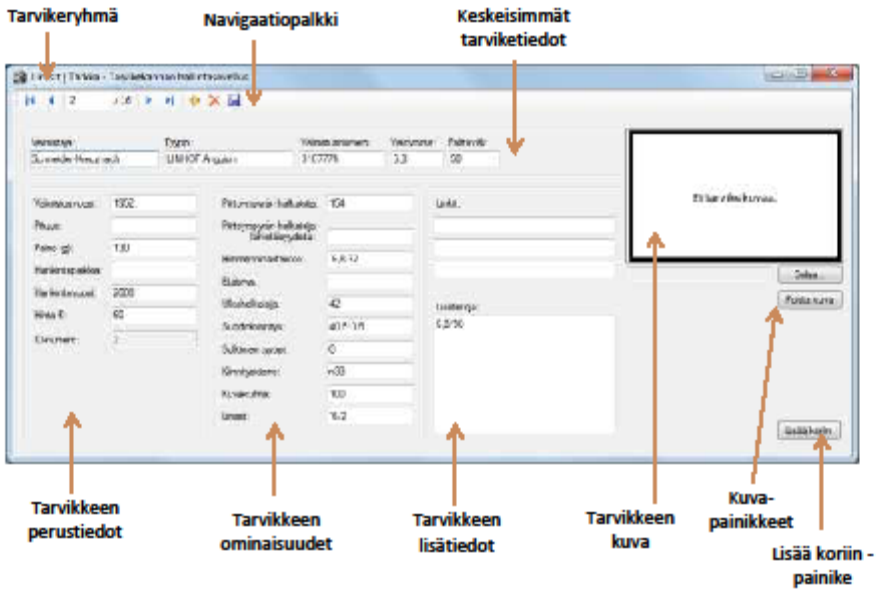


Huomaa, että tarviketietojen tai korien hallintaikkunoiden sulkeminen ei sulje koko sovellusta!

2 Tietojen hallinta

2.1 Tarviketietojen hallintaikkunan rakenne

Tarviketietojen hallintaikkunat on järjestetty samoin tarvikeryhmästä riippumatta.

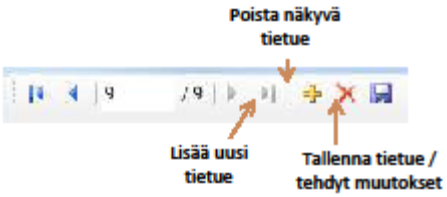


2.2 Tietojen selaus

Tietoja selataan navigaatiopalkin kautta.

Tietoja voi selata myös syöttämällä tietueen numero selauspainikkeiden välissä olevaan tekstikenttään.

2.3 Tarviketietojen lisäys, muokkaus, poisto ja tallennus



2.3.1 Uuden tarviketiedon lisäys

1. Klikkaa Lisää uusi -painiketta navigaatiopalkissa
2. Kirjaa tarvikkeen tiedot tarvikekortille
3. Tallenna: Navigaatiopalkki → Tallenna-painike

Huomaa, että prototyyppi ei huomauta tallentamattomista tiedoista. Muista siis tallentaa tiedot, ennen toiseen tietueeseen siirtymistä tai ohjelman sulkemista!

2.3.2 Tarviketietojen muokkaus

1. Jos tietokenttä on ennestään tyhjä, kirjoita uusi tieto kenttään.

   tai
2. Jos tietokentässä on jo tieto, poista se ja kirjoita kenttään uusi tieto.
3. Tallenna: Navigaatiopalkki → Tallenna-painike

2.3.3 Tarviketietojen poisto

1. Klikkaa navigaatiopalkin Poista tietue -painiketta.

Huomaa, että poistat aina näkyvän tietueen.

2.3.4 Tarviketietojen tallennus

1. Klikkaa navigaatiopalkin Tallenna tietue -painiketta.

Huomaa, että prototyyppi ei huomauta tallentamattomista tiedoista. Muista siis tallentaa tiedot, ennen toiseen tietueeseen siirtymistä tai ohjelman sulkemista!

2.4 Linkkien avaaminen

1. Tuplaklikkaa linkkikenttää.

Sovellus avaa linkit oletusselaimessa.

2.5 Kuvatoiminnot

2.4.1 Kuvan lisäys

1. Klikkaa Selaa... -painiketta
2. Selausnäkymän avauduttua etsi koneeltasi kuva, jonka haluat tarviketietoihin liittää.
3. Valitse kuva.
4. Klikkaa Avaa-painiketta.
5. Tallenna: Navigaatiopalkki → Klikkaa Tallenna-painiketta

Huomaa, että sovellus hyväksyy .jpg- ja .jpeg-muotoiset kuvatiedostot. Muunlaisia tiedostomuotoja ei hyväksytä ja niitä lisättäessä sovellus antaa huomautuksen.

2.4.2 Kuvan tallennus

1. Navigaatiopalkki → Klikkaa Tallenna-painiketta

2.4.3 Kuvan poistaminen

1. Klikkaa Poista kuva -painiketta
2. Tallenna tehty muutos: Navigaatiopalkki → Klikkaa Tallenna-painiketta

2.4.4 Kuvan vaihtaminen

Katso:

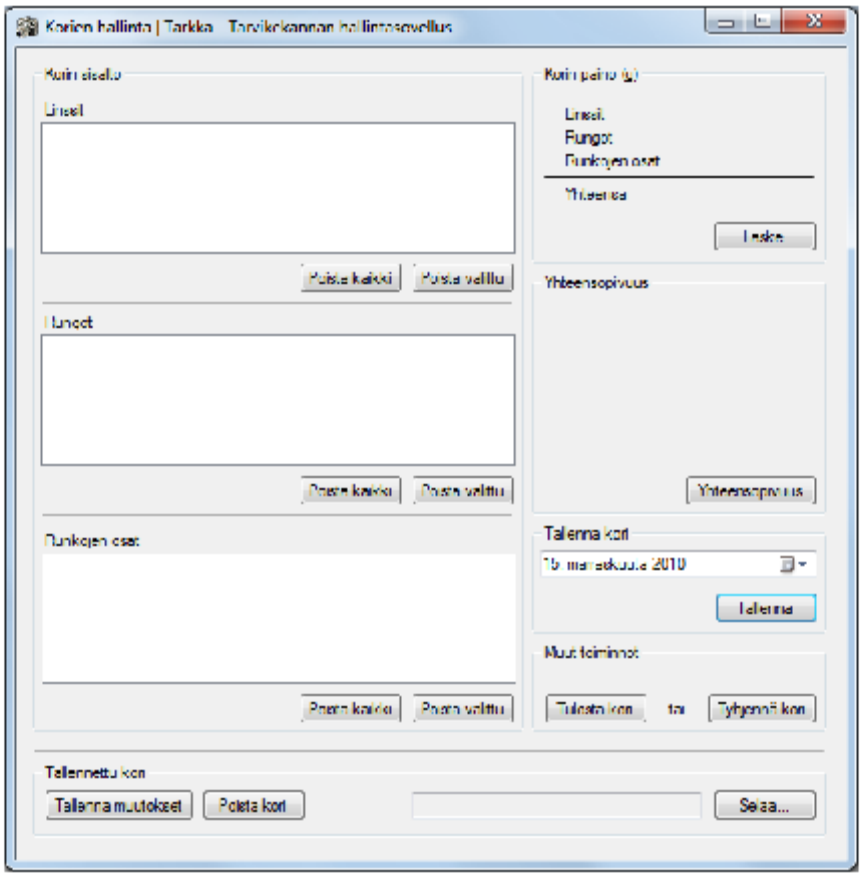
- 2.4.1 Kuvan lisäys
- 2.4.2 Kuvan tallennus

2.4.5 Kuvan suurentaminen

1. Tuplaklikkaa kuvaa tarviketietojen hallintaikkunassa.

3 Korien hallinta

3.1 Korien hallintaikkunan rakenne



3.2 Korin sisällön hallinta

3.2.1 Tarvikkeen lisääminen koriin

Tarvike lisätään koriin tarviketietojen hallintaikkunan kautta Lisää koriin -painikkeella. Painikkeen klikkaaminen avaa korien hallintaikkunan.

Huomaa, että kori tyhjenee, mikäli suljet hallintaikkunan. Prototyyppi ei huomauta tallentamattomista tiedoista. Muista siis tallentaa korin tiedot, ennen korin tai ohjelman sulkemista!

3.2.2 Tarvikkeen poistaminen korista

Tarvikkeita pystyy poistamaan kolmella tavalla.

3.2.2.1 Yhden tarvikkeen poistaminen

1. Valitse poistettava tarvike listasta.
2. Klikkaa tarvikeryhmän alapuolella olevaa Poista valittu -painiketta.

3.2.2.2 Tarvikeryhmän poistaminen

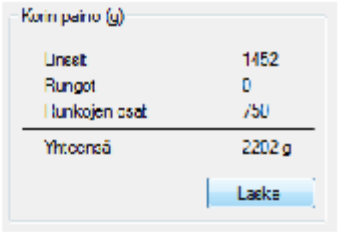
1. Klikkaa tarvikeryhmän alapuolella olevaa Poista kaikki -painiketta.

3.2.2.3 Koko korin tyhjentäminen

1. Klikkaa oikeassa laidassa olevaa Tyhjennä kori -painiketta.

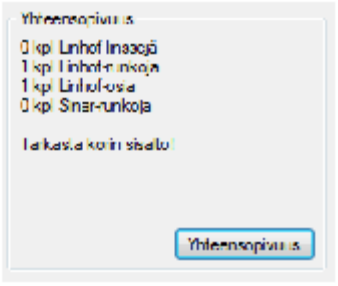
3.2.3 Korin painon laskeminen

Korin paino lasketaan klikkaamalla Laske-painiketta. Paino näytetään grammoina sekä ryhmäkohtaisesti että koko korin yhteenlaskettuna painona.



3.2.4 Tarvikkeiden yhteensopivuuden tarkastaminen

Koriin lisättyjen tarvikkeiden yhteensopivuuden voi tarkastaa klikkaamalla Yhteensopivuus-painiketta.

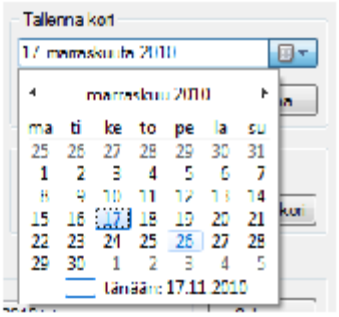


Prototyyppi tarkastaa Sinar- ja Linhof-tarvikkeiden määrän ja huomauttaa, mikäli on syytä tarkastaa korin sisältö.

3.2.5 Korin tallentaminen

1. Valitse kalenterinäkymästä haluamasi päivä.
2. Klikkaa Tallenna-painiketta.

Huomaa, että prototyypissä yhdelle päivälle voidaan tallentaa vain yksi kori. Prototyyppi tallentaa korit tekstitiedostona kansioon C:\Tarkka_TallennetutKorit\.

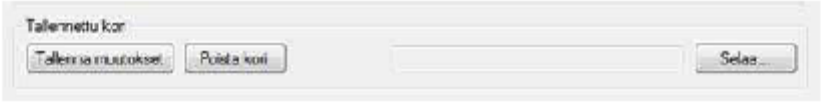


3.2.6 Korin tulostaminen

1. Klikkaa Muut toiminnot -ryhmässä olevaa Tulosta kori -painiketta.

3.2.7 Tallennetun korin avaaminen

1. Klikkaa Selaa... -painiketta.
2. Avautuvassa selausikkunassa on näkyvillä kansio, johon koritiedostot on tallennettu.
3. Valitse haluamasi kori ja klikkaa Avaa-painiketta.



3.2.8 Tallennetun korin muokkaaminen

Voit tehdä tallennettuun koriin muutoksia avaamalla korin hallintaikkunassa. Tallennetun korin muokkaaminen tapahtuu samoin kuin tallentamattomankin korin muokkaaminen. Katso:

- 3.2.7 Tallennetun korin avaaminen
- 3.2.1 Tarvikkeen lisääminen koriin
- 3.2.2 Tarvikkeen poistaminen korista
- 3.2.3 Korin painon laskeminen
- 3.2.4 Tarvikkeiden yhteensopivuuden tarkastaminen
- 3.2.6 Korin tulostaminen

3.2.9 Tallennetun korin uudelleen tallentaminen

Tallennettu kori voidaan tallentaa kahdella tavalla:

1. Klikkaamalla Tallenna muutokset -painiketta Tallennettu kori-ryhmässä.

tai

2. Tallentamalla kori uudella nimellä. Katso: 3.2.5 Korin tallentaminen

3.2.10 Tallennetun korin poistaminen

1. Avaa haluamasi tallennettu kori. Katso: 3.2.7 Tallennetun korin avaaminen
2. Klikkaa Tallennettu kori -ryhmässä olevaa Poista kori -painiketta.

Huomaa, että sovellus ei kysy varmistusta korin poistamiselle.