Marko Päivärinta

Tuotedokumenttien hallintajärjestelmän toteutus

NAMMO LAPUA OY:lle

Opinnäytetyö
Syksy 2011
Tekniikan yksikkö, Seinäjoki
Tietojenkäsittelyn koulutusohjelma
Elektroninen liiketoiminta

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö:	Tekniikan yksikkö
Koulutusohjelma:	Tietojenkäsittelyn koulutusohjelma
Suuntautumisvaihtoehto:	Elektroninen liiketoiminta
Tekijä:	Marko Päivärinta
Työn nimi:	Tuotedokumenttien hallintajärjestelmän toteutus NAMMO LAPUA OY:lle
Ohjaaja:	Erkki Koponen
Vuosi: 2011	Sivumäärä: 63 Liitteiden lukumäärä: 5

Opinnäytetyön tarkoituksena oli kuvata suunnitteluprosessia ja toteustapoja, joilla Nammo Lapua Oy:n käyttöön rakennettiin uusi tuotedokumenttien hallintajärjestelmän vanhan tilalle. Näkökantana on liiketoiminnan kehittäminen ja se, kuinka vanhojen järjestelmien korvaaminen uusilla useimmissa tapauksissa tehostaa yrityksen toimintaa. Koska järjestelmän toteuttamisen yhteydessä ohjelmointi ja valmiiden ratkaisujen räätälöinti tarvittavaan muotoonsa oli keskeisessä osassa, opinnäytetyössä käydään läpi myös ohjelmistotuotannon yleistä teoriaa ja järjestelmästä on liitetty mukaan useita ohjelmakoodiesimerkkejä.

Työssä kuvataan ensin hieman järjestelmäsuunnittelun teoriaa ja dokumenttien hallinnan yleisiä ongelmia. Sen jälkeen käydään läpi yrityksessä olleen vanhan järjestelmän ongelmakohtia ja suunnitteluprosessin aikana tehtyjä suunnitelmia. Lopuksi kuvataan uuden järjestelmän rakennetta ja toimintaa sopivien esimerkkien, kuvien ja sanallisen selostuksen avulla.

Tuotedokumenttien hallintajärjestelmä toteutettiin Borland Delphillä ja sen versiolla 6.0. Muita tärkeitä käytettyjä teknisiä ratkaisuja olivat SQL Server, Active Directory ja kaksi ostettua Delphin komponenttia. Näiden avulla luotiin järjestelmä, jonka avulla tuotedokumenttien tuotetiedot ja tyhjät dokumenttipohjat olivat molemmat tallennettu tietokantaan. Näin järjestelmä mahdollistaa nykyaikaisen ja tehokkaan dokumenttien hallinnan.

Järjestelmälle asetetut tavoitteet saavutettiin tärkeimmiltä osiltaan ja mahdollinen jatkokehitys on myös tarpeen mukaan toteutettavissa. Valmistunut järjestelmä palvelee nyt tuotekehitysosaston tarpeita ja sen avulla tuotedokumenttien käsittelyssä olleet ongelmat ja epäkohdat on saatu hallintaan.

Asiasanat: dokumenttien hallinta, tuotedokumentti, tietokantasovellus, sovelluksen kehittäminen, Delphi

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology
Degree programme: Business Information Technology
Specialisation: Electronic Business

Author: Marko Päivärinta

Title of the thesis: Creation of product document management system for Nammo Lapua Ltd

Supervisor: Erkki Koponen

Year: 2011 Number of pages: 63 Number of appendices: 5

The purpose of this thesis was to describe the specification process, and techniques used to construct a new document management system in Nammo Lapua Ltd. The point of view is mainly in development of business processes, and describing how replacing old systems usually makes processes more efficient. Modifying commercial software components as specified parts of the system was an important part of system development, therefore system development theory and examples of software code were included.

First, the theory and description of general problems in document management were presented. Next, the biggest problems of the old document management system and the plans created for the new system were described. Finally, the structure of the system was described literally and by examples and pictures.

The document management system was constructed with Delphi 6.0. Other key techniques used were SQL Server, Active Directory, and two commercial Delphi-components. Using these components and techniques the management system saves documents into database. These methods of saving data and the database structure used makes the system work efficiently.

The goals planned for the system were mainly all achieved and future development is also possible. All the biggest problems of managing product documents are now under control.

Keywords: document management, product document, database application, software development, Delphi

SISÄLTÖ

Käytetyt termit ja lyhenteet

Delphi Delphi on Borlandin tekemä Object Pascal -kieleen perustuva graafinen ohjelmankehitysympäristö ja ohjelmointikieli.

SQL Server SQL Server on Microsoftin kehittämä relaatiotietokanta palvelinohjelmisto.

Active Directory Active Directory (AD) on Microsoftin Windows-toimialueen käyttäjätietokanta ja hakemistopalvelu, joka sisältää tietoa käyttäjistä, tietokoneista ja verkon resursseista.

PDF Portable Document Format on Adoben kehittämä ohjelmistoriippumaton, siirrettävä tiedostomuoto. Sitä käytetään pääasiallisesti sähköiseen julkaisemiseen, tulostamiseen ja painamiseen.

SQL Structured Query Language on IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä. Käytännössä kaikki relaatiotietokannat ymmärtävät SQL-kieltä.

Tietokanta Tietokanta on tietotekniikassa käytetty termi tietovarastolle. Se on kokoelma tietoja, joilla on yhteys toisiinsa.

Sovellus Tietokoneohjelma (ohjelma, sovellus) on joukko ennalta laadittuja käskyjä, joita seuraten tietokone suorittaa sille valmistellun tehtävän. Useasta ohjelmasta koostuvaa yhtenäistä kokonaisuutta voidaan kutsua myös ohjelmistoksi.

Stored Procedure Tallennettu proceduuri on aliohjelma, joka on tietokantaa käyttävien ohjelmien käytettävissä. Se on tallennettu tietokantaan.

(Anttila, J. 2001; Heikkinen, A.1994; Hovi, A. 2004; Swan, T. 1999; Microsoft. 2011)

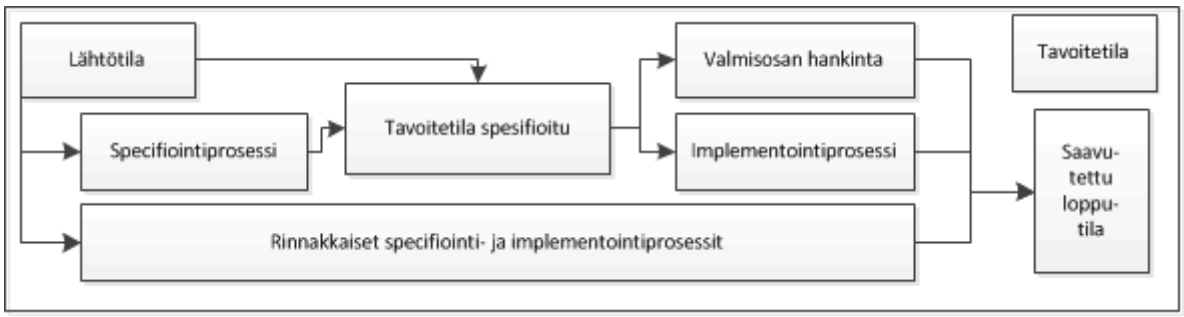
Kuvioluettelo

Esimerkkiluettelo

1 JOHDANTO

Nykyajan tietoyhteiskunnassa jokainen menestyvä ja vähänkin isompi yritys tarvitsee toimivan dokumenttien hallinnan. Dokumenttien monimutkaistuessa ja tietomäärän lisääntyessä niitä on mahdotonta hallita tehokkaasti, jos käytössä ei ole jonkinlaista nykyaikaista dokumenttien hallintajärjestelmää. Nammo Lapua Oy:ssä dokumenttien hallinta on myös osa yrityksen jokapäiväistä toimintaa. Patruunoiden ja ammustarvikkeiden tuotantolaitoksena sen tuotannon ja laadun takeena pyörii päivittäin useita eri dokumentteja, joiden sisältöä tarvitaan niin tuotannossa, varastossa, myynnissä kuin valmistettujen tuotteiden testauksessakin. Kaikkien toimintojen sujuvuuden ja jatkuvuuden varmistamiseksi toimiva dokumenttien hallinta on tärkeää. Sen avulla voidaan olla varma, että kukin osasto ja työntekijä saavat aina viimeisimmän ja ajantasaisen tiedon haetuista tiedoista ja dokumentista.

Tämän työn tarkoituksena on kuvata prosessia, jolla Nammo Lapua Oy:lle rakennettiin toimiva tuotedokumenttien hallintajärjestelmä. Työssä kuvataan myös yleisiä dokumentinhallinnan periaatteita, siinä usein esiintyviä ongelmia ja niiden ratkaisumalleja. Työssä ei kuvata dokumenttienhallintajärjestelmän rakentamista alusta loppuun. Tätä työtä voidaan kuitenkin käyttää esimerkkinä ja uusien ideoiden lähteenä suunniteltaessa samanlaista järjestelmää joko täysin uutena tai jonkun vanhemman järjestelmän tilalle. Työn tarkoituksena ei ole käydä läpi koko järjestelmän ominaisuuksia ohjelmointitasolla, vaan antaa yleiskuva siitä, minkälainen järjestelmä on mahdollista rakentaa valmiiden komponenttien ja järjestelmien sekä itse tehdyn räätälöinnin avulla.



Kuvio 1. Järjestelmän toteutuksen eteneminen (Haikala & Märijärvi, 2004, 108)

Kuvoissa 1 on kaaviona esitetty kuinka lähtötilasta voidaan eri reittejä pitkin edetä tavoitetilaan. Tässä kyseisessä projektissa edettiin kuvion 1 suuntaviivoja noudattaen eli alkuun määriteltiin lähtötilavaiheessa nykyinen tilanne, vanha järjestelmä sekä vanhan järjestelmän ja nykyisen tilanteen puutteet ja ongelmat. Tämän jälkeen määriteltiin tavoitetila ja sen toteutukselle asetetut tavoitteet. Tavoitetilaan pääsemiseksi aloitettiin toteutusprosessi, jonka aikana järjestelmään ostettiin sekä valmisosia että räätälöitiin niiden toimintaa omien järjestelmien osana. Lopputuloksena päädyttiin lopputilaan eli toteutettuun tuotedokumenttien hallintajärjestelmään.

Luvussa kaksi käydään läpi yleisiä dokumenttien hallintaan liittyviä asioita sekä millaisia dokumentteja yrityksissä nykyään on ja minkälaisia ongelmia niiden hallintaan yleensä liittyy. Luvussa kolme käydään läpi Nammo Lapua Oy:n tuotedokumentteja yleisellä tasolla, niiden hallintaa ennen toteutettua uutta dokumenttien hallintajärjestelmää sekä niitä ongelmia, joita ennen järjestelmän toteutusta dokumenttien hallinnassa yleensä ja useimmiten oli. Lisäksi käydään läpi niitä ominaisuuksia, joiden avulla uuden järjestelmän haluttiin ratkaisevan aiemmin esiintyneitä ongelmia. Luvussa kolme kerrotaan järjestelmän toteutuksesta. Aluksi käydään läpi projektin organisaatio ja hieman yleistä järjestelmien suunnittelun teoriaa. Lisäksi esitellään lyhyesti kehitysvälineenä käytetty Borland Delphi. Tämän jälkeen luvussa käydään läpi järjestelmä ensin yleisellä tasolla kuvaten rakennetta ja sen toteuttamisvaiheita kokonaisuudessaan, sitten järjestelmän rakenne kuvataan yksityiskohtaisemmin moduuli moduulilta. Luvussa viisi käydään vielä läpi johtopäätökset ja valmiin järjestelmän arviointi eli millaisia asioita järjestelmän avulla saavutettiin ja kuinka projektissa kokonaisuudessaan onnistuttiin.

2 DOKUMENTTIEN HALLINTA JA HALLINTAJÄRJESTELMÄT

Nykyisin yrityksissä tuotetaan dokumentteja yhä vain enemmän. Perinteiset paperidokumentit ovat edelleen tärkeitä, mutta niiden lisäksi syntyy lukematon määrä sähköpostiviestejä, www-sivuja, muistioita, laskentataulukkoja sekä piirustuksia tietokoneiden kiintolevyille. Tähän vaikuttaa paljon myös se, että tietoliikenne on erittäin tehokasta sähköpostin ja internetin välityksellä. Tiedon siirtäminen ja jakelu paikasta toiseen on siis helppoa, joten tämä kasvattaa helposti myös epäolennaisen tiedon määrää. Samalla odotetaan, että tieto on käytettävissä aina, ajasta ja paikasta riippumatta. Tämä osaltaan lisää tiedon tulvaa ja nykyisin suurimpia ongelmia onkin löytää olennainen ja ajantasainen tieto kaiken epäolennaisen joukosta (Anttila 2001, 1.)

2.1 Dokumentti, mikä se on?

Dokumentti on yleensä jokin asiakokonaisuus, joka on tarkoitettu ihmisten luettavaksi. Sen määritelmä voisikin olla esimerkiksi, että se on ihmisen käsiteltäväksi tarkoitettu tieto. Perinteisesti dokumentit ovat olleet aina paperimuodossa, mutta nykyisin dokumenteista suurin osa on sähköisiä dokumentteja. Termit elektroninen, sähköinen ja digitaalinen tarkoittavat tässä yhteydessä kaikki samaa eli sähköistä dokumenttia, joka on tallennettu tietokoneen ymmärtämässä muodossa (Anttila 2001, 1.)

Mikä tahansa tiedosto ei kuitenkaan ole dokumentti. Dokumenteista puhuttaessa dokumentin muodostavat itse tallennettu tiedosto ja kaikki sitä kuvaavat tiedot, jotka on tallennettu sen yhteydessä. Nämä tiedot kertovat yksityiskohtaisemmin ja nopeammin, mitä tietoja käsiteltävänä oleva dokumentti sisältää, kuten esimerkiksi sen, että avattu dokumentti on edellisestä palaverista tehty muistio tai jotain muuta vastaavaa. Dokumentti voi olla myös useamman tiedoston ja niitä kuvaavien tietojen yhdistelmä, jolloin kaikki edellä mainitut tiedot yhteensä muodostavat yksittäisen kokonaisuuden (Anttila 2001, 2.)

2.2 Perinteisen dokumenttienhallinnan ongelmia

Aikaisempaan verrattuna dokumentteja tuotetaan nykyisin paljon enemmän kuin ennen. Tämä johtuu siitä, että verrattuna aivan ensimmäisiin tietojärjestelmiin, nykyajan sovellukset ovat paljon helppokäyttöisempiä, tehokkaampia ja monipuolisempia. Toisaalta lainsäädännön muuttuminen sekä uudenlaiset standardit ja ohjeet vaativat usein tarkempaa dokumentaatiota. Niinpä uusien dokumenttien luominen ja muokkaaminen on toimistotyössä jokapäiväistä työtä (Anttila 2001, 3.)

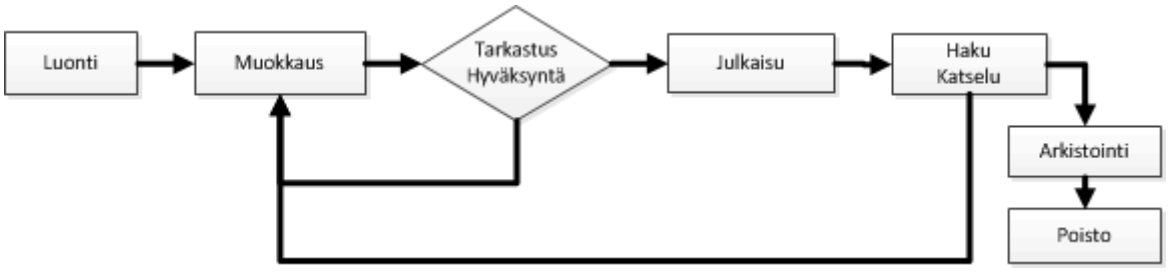
Mitä enemmän dokumentteja on, sitä työläämpää niitä on hallita. Dokumenttien työläämpi hallinta tarkoittaa yleensä myös sitä, että mahdollisia virhetilanteita sattuu useammin. Usein on helpompaa tehdä joku pieni dokumentti alusta uudestaan, jos jo aiemmin tehdyn version löytäminen on liian vaikeaa. Dokumentteja on helppo myös tuhota vahingossa tai tallentaa joku vanhempi versio uudemman päälle, jos dokumenteista on monia eri versioita eri paikoissa. Osa dokumenteista voi olla tiedoiltaan lisäksi niin arvokkaita, että tietojen hukkumisesta koituvat menetykset voivat olla hyvinkin suuria. Jotta dokumenttien luominen ja muokkaaminen olisi mahdollisimman tehokasta, dokumenteista pitäisi olla löydettävissä viimeisin versio helposti ja nopeasti. Näin dokumentteja muutettaessa aikaa ei kulu turhaan oikean ja ajantasaisen tiedon etsimiseen tai jopa täysin uuden dokumentin luomiseen (Anttila 2001, 3.)

USA:ssa on tehty aiemmin tutkimuksia, joiden mukaan huono ja puutteellinen dokumenttien hallinta voi aiheuttaa jopa noin 170 000 euron kustannukset vuodessa 100 hengen yrityksessä. Tämä aiheutuu suurimmaksi osaksi siitä työstä, joka tulee kadonneiden dokumenttien etsimisestä ja uudelleen kirjoittamisesta, jos alkuperäisiä versioita ei enää löydetä. Myös Suomessa yritykset kohtaavat samoja ongelmia, jos dokumenttien hallinta on puutteellista. Aikaa on helppo saada kulumaan turhaan asiakirjojen etsimiseen, kopiointiin, postitukseen sekä papereiden järjestämiseen ja arkistointiin. Toimivan dokumenttien hallintajärjestelmän avulla tämä aika voidaan minimoida ja vapauttaa sitä edelleen johonkin muuhun tarpeelliseen ja tuottavaan työhön (Heikkinen 1994, 51.)

2.3 Dokumenttien hallinta sähköisesti

Dokumenttien tehokkaaseen hallintaan kuuluu olennaisena osana nykyaikainen dokumenttien hallintajärjestelmä. Järjestelmä pitää yleensä sisällään sekä tiedostot että niitä kuvaavat ominaisuustiedot. Nämä tiedot tallennetaan tietokantaan, jonne voidaan tallentaa myös dokumentteihin liittyvät tiedostot, jolloin kaikki dokumentteihin liittyvät tiedot ovat yhdessä ja samassa paikassa. Järjestelmän rakenne mahdollistaa tietojen hakemisen tehokkaasti sekä valittujen tietojen esittämisen käyttäjälle selkeästi ja joustavasti. Dokumentteja voidaan hakea useilla eri hakuehdoilla sekä hakusanoja voidaan kohdistaa myös itse dokumentin sisältämään tietoon, mikä puolestaan myös helpottaa oikean ja viimeisimmän dokumentin löytämistä (Anttila 2001, 4.)

Yksi hallintajärjestelmien tärkeimmistä ominaisuuksista on se, että järjestelmä huolehtii dokumenttien versioinnista automaattisesti. Tämän lisäksi hallintajärjestelmät pitävät myös huolta siitä, että kaksi eri käyttäjää ei pääse muokkaamaan samaa dokumenttia yhtä aikaa. Dokumenttien katselua ja muokkausta varten on yleensä mahdollista myös määritellä erilaisia käyttöoikeuksia, jolloin eri käyttäjillä on eritasoiset oikeudet tehdä muutoksia dokumentteihin. Lisäksi hallintajärjestelmistä löytyy yleensä automatiikkaa, joka huolehtii erilaisista dokumentteihin liittyvistä työvirroista. Tällaisia ovat esimerkiksi dokumenttien tarkastus, hyväksyntä, julkaisu ja jakelu. Mikään ei estä myös sitä, että dokumenttien hallintajärjestelmän avulla hallitaan edelleen paperilla tulostettuna olevia dokumentteja. Tällöin dokumenttien ominaisuustietojen lisäksi järjestelmään tallennetaan paperisen dokumentin sijainti, jolloin kyseinen dokumentti on helppo ja nopea löytää. Kuviossa 1 on Juha Anttilan kuvaus dokumentin elinkaaresta. Siinä on kuvattu kuinka dokumentti luodaan ja kuinka sitä muokataan tarkastus- ja hyväksyntämenettelyiden osana. Lisäksi elinkaareen kuuluvat julkaisu, haku, katselu ja loppuvaiheessa arkistointi ja poisto (Anttila 2001, 4-5.)



Kuvio 2. Dokumentin elinkaari (Anttila 2001, 5.)

3 TUOTEDOKUMENTIT NAMMO LAPUA OY:SSÄ

Erilaisia dokumentteja ja dokumenttien hallintajärjestelmiä on ollut ja tulee olemaan yrityksissä useita. Niin myös Nammo Lapua Oy:ssä. Tehtaalta löytyy useita osastoja varastosta ja tuotannosta aina myyntiin ja tuotekehitykseen asti. Erilaisen tiedon ja dokumenttien säilyttämisen ja hallinnan tarve onkin tärkeässä roolissa yrityksen toiminnassa. Dokumenttien koko ja informaatiosisältö vaihtelee laidasta laitaan, aina varaston osoitetarroista tuotekehityksen salaisiin kehityspäiväkirjoihin ja palaverimuistioihin asti. Lisäksi myyntiosastolta ja taloushallinnosta löytyy paljon tärkeätä tietoa ja kirjanpitoa myydyistä tuotteista aina henkilöstön palkkoihin ja työajan seurantaan liittyen. Voidaan siis sanoa, että dokumentteja on monia erilaisia, mutta mikään dokumentti ei ole tarpeeton, olkoon se kuinka pieni tai lyhyt tahansa. Omalla osastollaan jonkun yksittäisen dokumentin, pienenkin sellaisen, katoaminen, voi aiheuttaa hetken kestävän kaaoksen ja jopa toiminnan osittaisen tai täydellisen lamaantumisen.

3.1 Tuotedokumenttien hallinta vanhassa järjestelmässä ja sen ongelmat

Alkutilanteessa tuotedokumentteja varten ei ollut oikeastaan kunnollista hallintajärjestelmää tai sellaisena toimi Lotus Notes -ympäristössä tallennettuna oleva tietokanta. Tietokanta oli muodoltaan kuitenkin kuten sähköposti eli dokumentit voitiin tallentaa sinne yksittäisinä dokumentteina, vähän kuten sähköpostin liitteinä. Tallennettaessa voitiin ehkä lisätä muutamia avainsanoja dokumentin yhteyteen, mutta muunlaista etsi ja lajittele -toimintoa ei järjestelmässä oikein ollut. Koska doku-

mentit oli tallennettu järjestelmän sisälle omina palasinaan, niiden sisältöä ei voinut hakea, vertailla tai tutkia muuten kuin hakemalla yksittäinen dokumentti ja avaamalla se luettavaksi.

Dokumenttien hallinnassa olleet ongelmat olivat varsin ilmeisiä, koska jos dokumenttien tietoja haluttiin esimerkiksi vertailla tai tarkistaa, joutui niistä vastuussa oleva henkilö tekemään paljon manuaalista hakua ja vertailua ilman järjestelmän apua. Käytännössä tämä oli joko yksittäisten dokumenttien tulostusta paperille rinnakkain toistensa viereen tai jonkinlaista leikkaa liimaa puuhastelua eri ohjelmien välillä, ellei luottanut muistiinsa lukujen suhteen.

3.2 Uuden järjestelmän ominaisuudet ja tavoitteet

Uuden järjestelmän haluttiin ratkaisevan vanhan järjestelmän ongelmia eli tiedot haluttiin keskitetysti yhteen paikkaan niin, että niitä oli helppo hallita, muokata ja vertailla. Dokumentteihin liittyvät tiedostot, kuten kaaviot ja kuvat haluttiin myös tallentaa tietokantaan dokumentin yhteyteen. Lisäksi alusta alkaen kantavana ideana oli eriyttää dokumenttien ulkoasu ja varsinainen tietosisältö toisistaan. Näin ollen dokumenttien pohjia voitiin hallinnoida ja muokata erillään niiden sisältämien tietojen ja lukuarvojen muuttumatta.

Dokumenttien tiedot päätettiin tallentaa tietokantaan. Muita käyttökelpoisia vaihtoehtoja ei oikeastaan ollut, koska tietoja oli sen verran paljon, että niiden hallinnointi olisi Excelin tai Accessin avulla ollut hankalampaa. Koska järjestelmästä todennäköisesti tulisi useamman käyttäjän ympäristö, sellaista ei ole enää järkevää toteuttaa Excelin tai Accessin avulla. Täysverinen tietokanta mahdollistaisi myös tehokkaammat vertailu- ja ylläpito-operaatiot nyt ja varsinkin tulevaisuudessa, jos tietokannan sisältämät tiedot lisääntyvät suuresti tai jos järjestelmän yhteyteen päätetään rakentaa vielä jotain uutta.

Järjestelmän haluttiin mahdollistavan yhden ja saman käyttöliittymän kautta tuotetietojen kysely, uusien tuotetietojen lisääminen, olemassa olevien tuotetietojen muokkaaminen sekä niiden vertailu keskenään. Yhtenä päätavoitteena oli luoda

käyttöliittymään osio, jonka kautta tuotteiden tiedot voisi hakea eri hakuehdoilla rinnakkain näkyville, jolloin niitä olisi helppoa vertailla sekä tehdä tarvittaessa korjauksia ja muutoksia. Tavoitteena oli myös luoda käyttöliittymä, jonka avulla tuotedokumentin muokkaaminen olisi yhtä helppoa kuin tekstinkäsittely, jolloin järjestelmän käyttäminen onnistuisi mahdollisimman pitkälle ilman mitään erityistaitoja tai -oikeuksia. Myöhemmissä suunnittelupalavereissa järjestelmän ominaisuuksiin lisättiin vielä muun muassa käyttäjäoikeuksien hallinta ja pdf-tiedostojen luominen suoraan järjestelmän avulla. Järjestelmän suunnitelluista ominaisuuksista on suunnittelumuistio liitteenä 1.

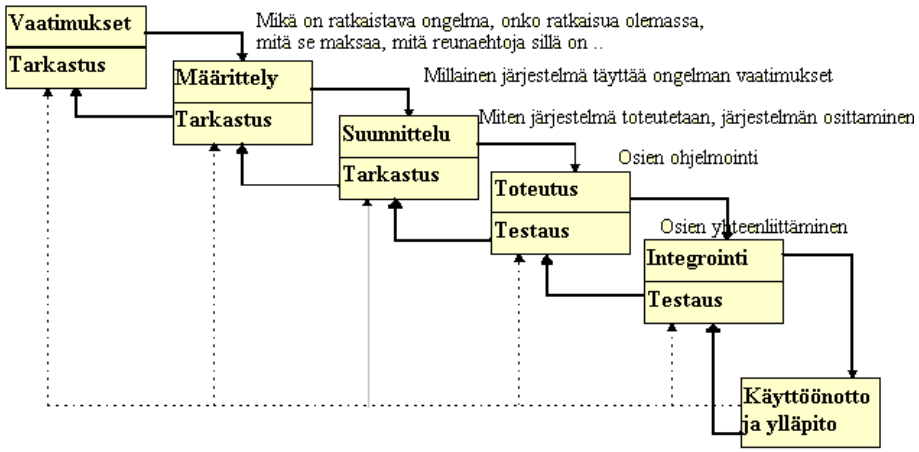
4 JÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

Järjestelmä suunniteltiin ja toteutettiin yhteistyössä Nammo Lapua Oy:n IT- ja tuotekehitysosaston kanssa. Suunnitteluun ja ideointiin ovat osallistuneet Nammo Lapuan tuotekehitys, IT-osasto, brändi- ja markkinointiasioista vastaavat henkilöt ja opinnäytetyön tekijä (kuvio 3). Ohjelmointi ja käytännön toteutus järjestelmän puolella on tehty opinnäytetyön tekijän toimesta. Tietokannan sisällön syöttäminen, tietojen ja dokumenttipohjien muotoilun hyväksyntä ja tarkastus on tehty yrityksen tuotekehityksen ja muiden asianosaisten yrityksen työntekijöiden toimesta. Tietokannan sisältö ja järjestelmän rakenne on muilta osin, kuin se tässä työssä on selostettu, Nammo Lapua Oy:n omaisuutta ja salassa pidettävää informaatiota.

Kuvio 3. Projektin organisaatio

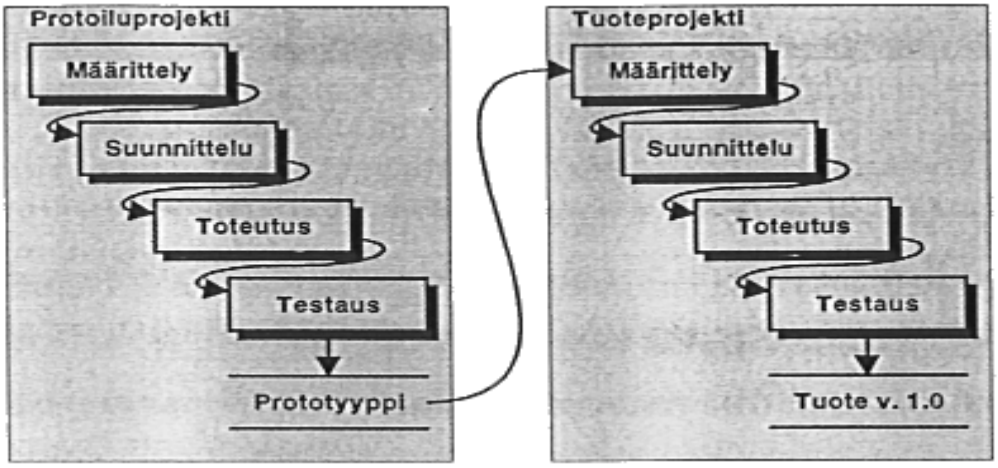
4.1 Tietojärjestelmien suunnittelusta

Tietojärjestelmien suunnitteluun kuuluvat olennaisina osina yleensä ainakin seuraavat osat: määrittely, suunnittelu, ohjelmointi ja testaus. Lisäksi suunnitteluprojekteihin liittyy koko suunnitteluvaiheen ja ohjelman käyttöajan kestäviä muita suunnitteluprosesseja tukevia toimintoja. Näistä tärkeimpiä ovat laadunvarmistus, tuotteenhallinta ja dokumentointi. Isoimmissa projekteissa muutamat muut tukitoiminnot saattavat vielä lisäksi olla omina toimintoinaan, kuten esimerkiksi vaatimusten- ja riskienhallintana. Pienemmissä projekteissa ne sisältyvät muiden toimintojen sisältöön (Haikala & Märijärvi, 2004, 35-36.)



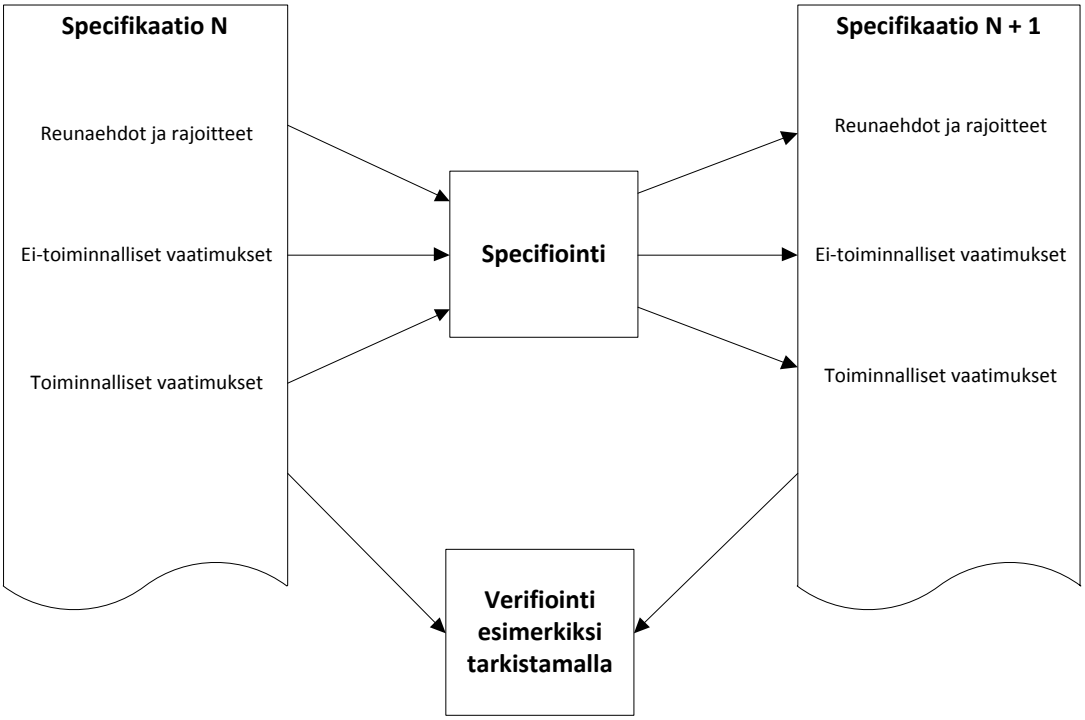
Kuvio 4. Vesiputousmalli (Haikala & Märijärvi, 2004, 36)

Tietojärjestelmien suunnittelussa käytetyistä vaihejakomalleista yleisimmin käytetty on ns. vesiputousmalli. Kaaviokuva vesiputousmallista on esitetty kuviossa 4. Vesiputousmallissa määrittelyvaihetta edeltää yleensä esitutkimus ja tarvekartoitusvaihe. Vaiheista on olemassa eri muunnelmia, mutta alun esitutkimuksen jälkeen seuraavaksi suoritetaan yleensä määrittely-, suunnittelu-, ja toteutusvaiheet. Kaikkien vaiheiden jälkeen tai niiden aikana suoritetaan jatkuvaa laadunvarmistusta eli tehtyjä suunnitelmia ja päätöksiä arvioidaan katselmuksin, tarkastuksin ja testauksin. Näiden toimien tarkoituksena on kitkeä mahdolliset virheet pois järjestelmästä jo suunnittelu- ja testausvaiheen aikana, jolloin valmis järjestelmä toimisi ilman virheitä. Päävaiheiden jälkeen voidaan järjestää vielä katselmuksia, joissa todetaan tarvittavat työt tehdyksi ja ne myös dokumentoiduksi (Haikala & Märijärvi, 2004, 37.)



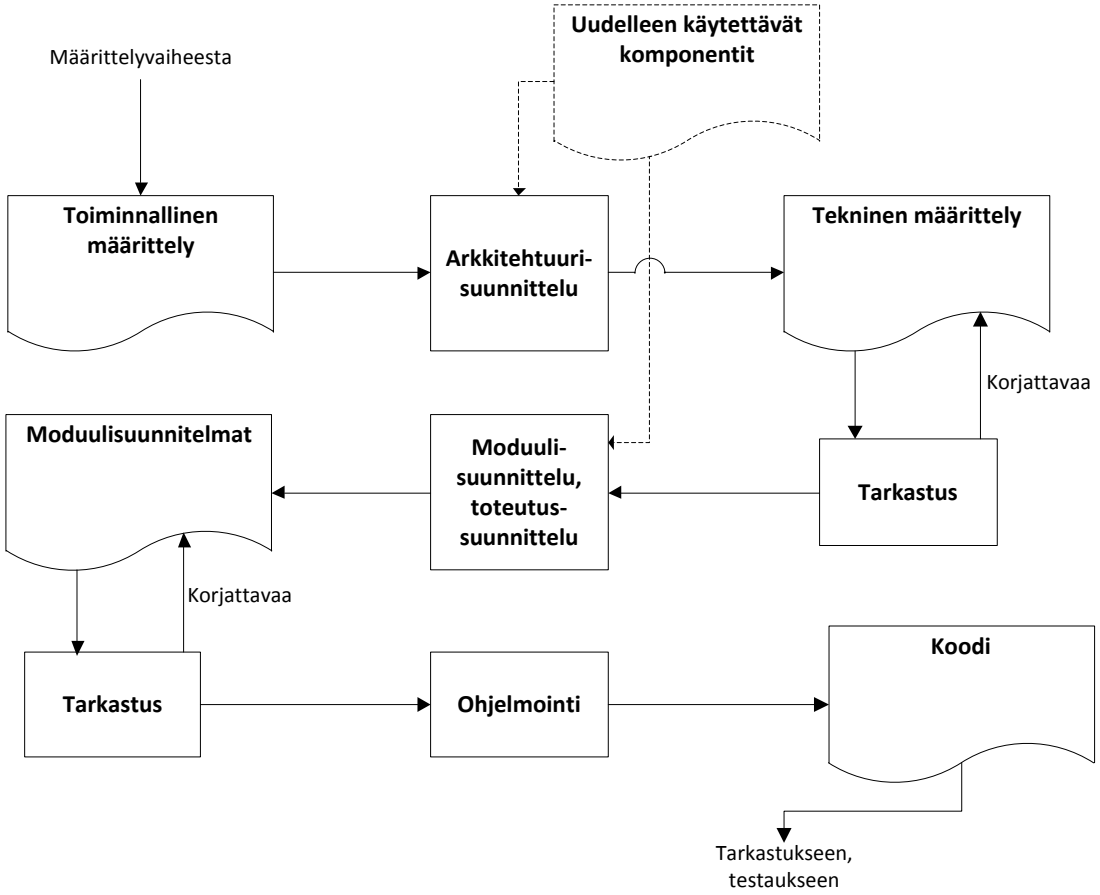
Kuvio 5. Protoilumalli (Haikala & Märijärvi, 2004, 42)

Muita paljon käytettyjä vaihejakomalleja edellisen vesiputousmallin lisäksi ovat muun muassa protoilumallit ja inkrementaaliset mallit. Protoilumallissa vaiheiden kulku etenee niin, että ennen varsinaisen tuotteen rakentamista tehdään yksi tai useampia protomalleja, joilla testataan ominaisuuksien toimintaa. Edellä kuvattu tapa edetä soveltuu erityisen hyvin jonkin uuden teknisen ratkaisun testaamiseen tai silloin, jos testataan tuotetta, josta on annettu epäselviä asiakasvaatimuksia. Protoilumallin periaatekuva on esitetty kuviossa 5 (Haikala & Märijärvi, 2004, 42.)



Kuvio 6. Spesifiointiprosessi (Haikala & Märijärvi, 2004, 64)

Spesifikaatiot ja onnistunut spesifiointivaihe on projektin ja järjestelmän toteutuksen kannalta hyvin tärkeä vaihe. Tässä vaiheessa tehdyt virheet tai mahdolliset puutteet määrittelyissä haittaavat ja hidastavat työn tekemistä myöhemmissä vaiheissa sekä saattavat pahimmillaan aiheuttaa hyvinkin kallista lisätyötä. Käytännössä järjestelmän määrittelyihin ja spesifiointidokumentteihin on hyvin vaikeata saada kaikki tarvittava. Niissä on yleensä aina jotain puutteellisuuksia tai monella tavalla ymmärrettäviä asioita, joiden sisältöä joudutaan miettimään uudestaan myöhemmissä projektin vaiheissa. Tyypillisiä mahdollisesti puuttuvia asioita ovat määrittelyt kuinka järjestelmän pitäisi toimia erilaisissa poikkeustilanteissa (Haikala & Märijärvi, 2004, 64-65.)



Kuvio 7. Suunnitteluprosessi (Haikala & Märijärvi, 2004, 82)

Asiakastarpeiden kartoituksen jälkeen, järjestelmien kehitys etenee suunnitteluprosessiin. Tämän prosessin tarkoituksena on muuttaa saadut toiveet ja tarpeet tekniselle kielelle ja kuvaukseksi järjestelmän toteutuksesta. Suunnitteluprosessin kaaviokuva on esitetty kuviossa 7. Suunnitteluvaiheet voidaan yleensä jakaa kahteen osaan: arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Näistä ensimmäisessä eli arkkitehtuurisuunnittelussa, suunniteltu järjestelmä jaetaan osiin ja osien rajapinnat määritellään. Moduulisuunnittelussa järjestelmän osat eli moduulit suunnitellaan tarkemmin ja niiden sisäinen rakenne määritellään. Tarkoituksena on lohkoa isompi järjestelmä niin pieniin osiin, että jonkun yksittäisen pienemmän osan toteutus on mahdollista yksittäiseltäkin työntekijältä. Moduulisuunnittelu voi olla myös sisällettynä muiden vaiheiden kanssa niin, että moduulin toteutus suunnittelusta testaukseen tehdään kaikki yhtenä vaiheena järjestelmän toteutuksen yhteydessä (Haikala, & Märijärvi, 2004, 81-82.)

4.2 Kehitysväline Delphi

Delphi on Borlandin tekemä Object Pascal -kieleen perustuva graafinen ohjelmankehitysympäristö ja ohjelmointikieli. Delphi mahdollisti julkaisuvuonnaan 1995 ensimmäisenä visuaalisen Windows-lomakkeiden editoinnin yhdistettynä olio-ohjelmointiin (Swan 1999, 4-11.)

Tällä hetkellä Delphin viimeisin versio on nimeltään Delphi XE2 ja se on julkaistu vuonna 2011. Sen ominaisuuksiin kuuluvat muun muassa 32- ja 64- bittisten ohjelmien tuki sekä mahdollisuus tehdä yhden sovelluskehitysalustan kautta ohjelmia PC- ja Mac-tietokoneille sekä iOS- mobiilijärjestelmiin, joihin kuuluvat muun muassa iPhone-puhelimet (Embarcadero. 2011.)

Delphissä on mukana iso joukko valmiita komponentteja, joiden avulla graafisen perusohjelman teko Windows-ympäristöön onnistuu helposti. Järjestelmän kehityksen kannalta tärkeimmässä asemassa ovat peruskomponenteista olleet erilaiset listat ja puunäkymät. Niiden avulla voi pienen rakentelun jälkeen tehdä selkeitä, monipuolisia, sekä paljon toimintoja sisältäviä ja samaan aikaan myös helposti käytettävissä olevia käyttöliittymiä Windows-ohjelmille. Edellisten tukena on käyttöliittymän rakennuksessa käytetty paljon vakiokomponenteista löytyviä teksti- ja otsikkokenttiä. Tietokantayhteys-komponentit tulevat myös vakiona Delphin mukana, joten tietokantapohjaisen järjestelmän rakentamiseen ei tarvitse yleensä ostaa mitään ylimääräisiä osia, ellei tietokanta ole jotain harvinaisempaa mallia (Swan 1999, 4-11.)

4.3 Järjestelmän toteutus

Nammo Lapua Oy:n tehtaalla Lapualla oli muita tietojärjestelmiä varten käytössä muun muassa SQL Server -palvelin ja aiemmin erilaisia ohjelmistoprojekteja oli toteutettu Delphi-ohjelmointikieltä käyttäen. Nämä kaksi työkalua valittiinkin alkuvaiheessa hyvin nopeasti käyttöön, koska niille oli olemassa olevat ja tarvittavat lisenssit sekä niiden ominaisuuksien arvioitiin riittävän reilusti suunnitellun järjestelmän käyttöön. Suunnittelu ja toteutus etenivät vesiputous- ja protoilumalleja

mukaillen eli eri protoversioiden välissä edettiin vesiputousmallin mukaisesti väliin määritellen, suunnitellen ja toteuttaen.

Tietokantapalvelimena SQL Server mahdollistaa järjestelmän tehokkaan käytön ja nykyaikaiset tietokantaominaisuudet. SQL Server on helppo asentaa tyhjään järjestelmään ja ottaa se siellä käyttöön. Sen lisäksi sitä on helppo ylläpitää. (Hovi, A. 2004.)

Koska järjestelmään haluttiin aika paljon toimintoja, nähtiin järkeväksi tutkia tiettyjen olemassa olevien komponenttien ominaisuuksia eikä tehdä kaikkea itse. Toisaalta, koska järjestelmä itsessään olisi omaa erityistä tarkoitustaan varten, valmista kokonaista hallintajärjestelmää ei nähty järkeväksi ostaa, koska sellaiset ovat yleensä kalliita tai niiden räätälöinti ja jatkokehitys tarpeita vastaaviksi tulisi myös joka tapauksessa maksamaan ja viemään aikaa.

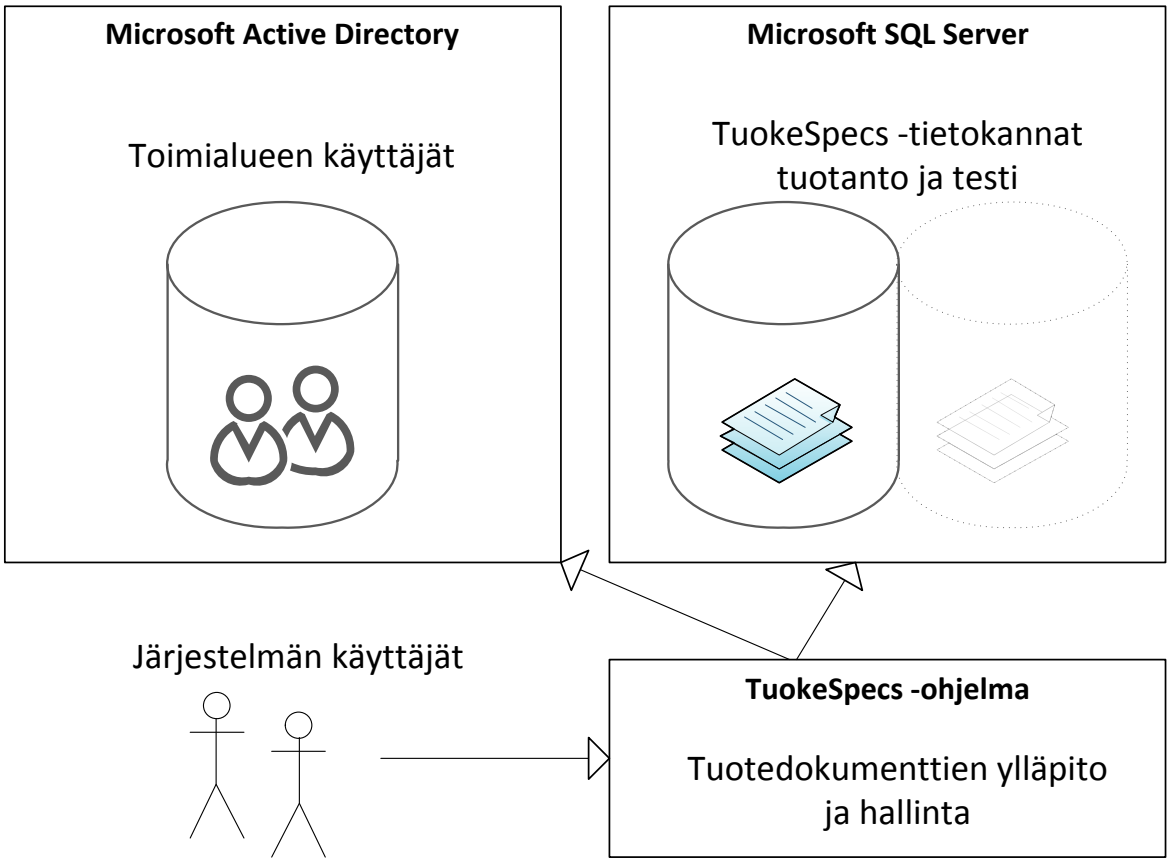
Muita käytettäviä työkaluja haettiin alkuvaiheessa jonkin aikaa, ja suunnitteluvaiheessa resursseja kului hiukan turhaankin erilaisten vaihtoehtojen testaamiseen, jotka myöhemmin osoittautuivat riittämättömiksi tai työläiksi käyttää. Vaihtoehtona harkittiin alkuvaiheessa erillistä xml-editoria tuotedokumenttien pohjien muokkaamista varten. Se hylättiin siksi, että se olisi jakanut järjestelmän kahteen erilliseen ohjelmaan ja siksi, että ominaisuuksiltaan sopivaa editoria oli yllättäen hiukan hankalaa löytää. Tämän jälkeen päätettiin huomio siirtää xml:stä valmiisiin Delphille tarjolla oleviin komponentteihin. Kaupallisten komponenttien joukosta valikoitui yksi eli RichViewEditor, joka myöhemmin osoittautui ominaisuuksiensa puolesta hyväksi valinnaksi. Myös käyttäjätuki sovelluskehitystä varten oli tällä komponentilla hyvä, mikä tuli muutaman ongelmatilanteen kautta todistettua myös käytännössä.

Järjestelmä suunniteltiin tulevan kokonaisuudessaan tuotekehityksen käyttöön, koska se tulisi sisältämään pääasiassa vain tuotekehityksen tarvitsemia tietoja ja dokumentteja. Lisäksi tietojen ylläpito tulisi olemaan täysin tuotekehitysosaston vastuulla. Yhteyksiä muihin tietojärjestelmiin sivuttiin myös alustavasti, koska tiettyjä tuotedokumentteihin liittyviä tietoja löytyy myös muista yrityksen tietojärjestelmistä. Tietokantayhteyksien avulla niitä voitaisiin tulevaisuudessa ehkä päivittää

tästä järjestelmästä käsin tai hakea jotain tietoja muista järjestelmistä tuotekehitysjärjestelmän käyttöön.

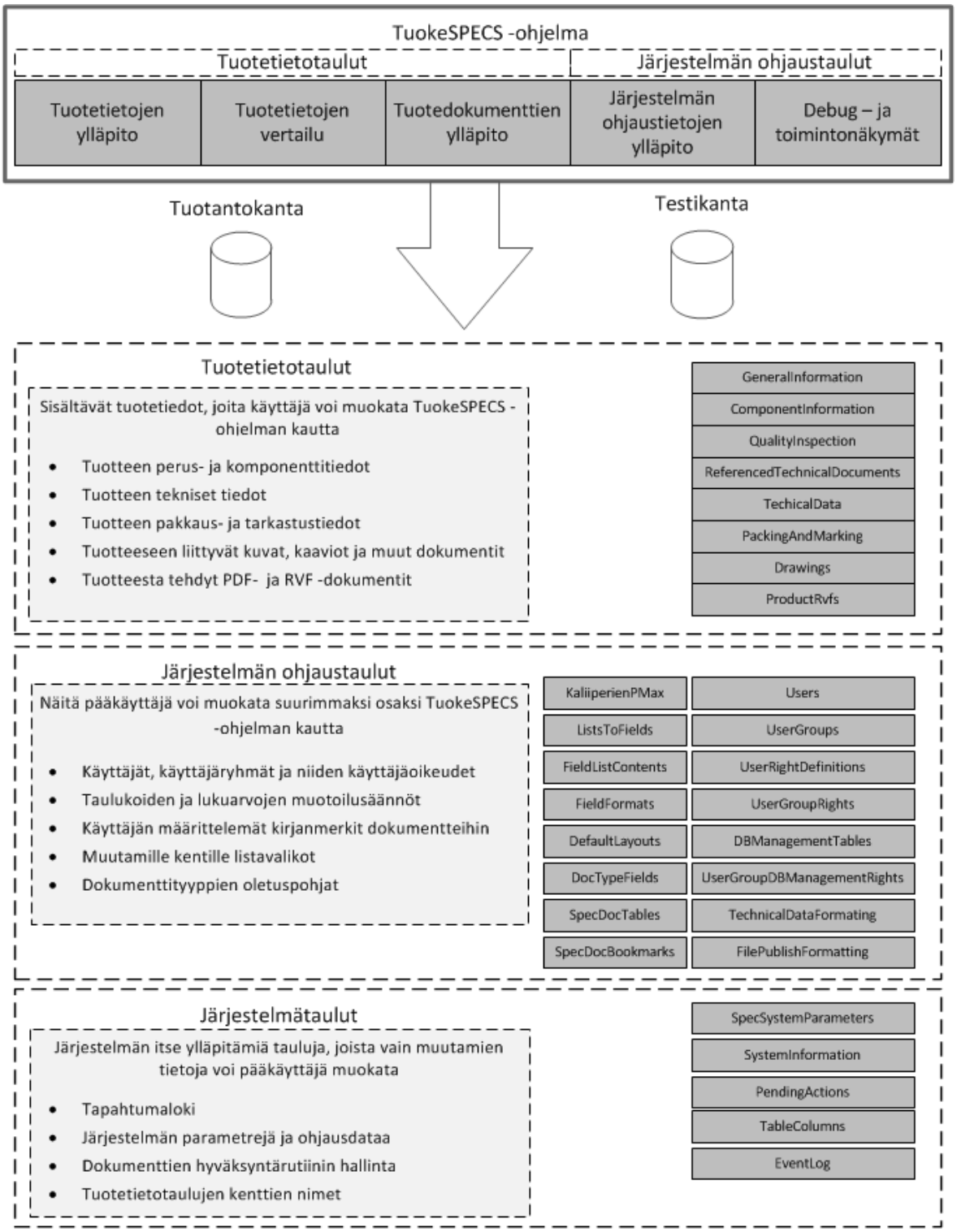
Järjestelmää varten ostettiin kaksi maksullista komponenttia. Niistä isompi ja järjestelmän kannalta eräs ydinkomponentti oli TRichView, jonka avulla tuotedokumentit näytettiin ruudulla käyttäjälle ja käyttäjä pystyi editoimaan niitä (TRichview. 2011). Pdf-tiedostojen luomista varten ostettiin TRichViewin kanssa yhteensopiva komponentti llPDFLib, jonka avulla luodut tuotedokumentit voidaan tallentaa järjestelmästä suoraan pdf-muotoon käyttämättä mitään ulkopuolista ohjelmaa (Llionsoft. 2011).

Tuotedokumenttien hallintajärjestelmä koostuu SQL Server-palvelimella olevasta tietokannasta ja tuotedokumenttien hallintaohjelmasta nimeltään TuokeSpecs *(*kuvio 8). Tietokantoja on kaksi, mutta ne ovat rakenteeltaan samanlaiset. Niin sanottu tuotantokanta pitää sisällään ajantasaiset tiedot sinne syötetyistä tuotteista ja niihin liittyvistä lukuarvoista, kuvista ja muista mahdollisista dokumenteista. Edellisen kanssa samanlainen tietokanta, testikanta, on rakenteeltaan täysin samanlainen, mutta sen sisältöä ei ylläpidetä samalla tavalla. Testikantaa voidaan käyttää tietojen ja järjestelmän ominaisuuksien testaamiseen ja sinne voidaan syöttää tuotteita, jotka eivät näy tuotantotietokannan puolella ollenkaan. Järjestelmän käyttäjät ovat Microsoft Active Directory-toimialueen käyttäjiä. Käyttäjänimet, joilla on oikeus käyttää järjestelmää on lisätty järjestelmän users-tauluun. Käyttäjien salasanat on tallennettu ja niitä hallinnoidaan Active Directoryssä (Mircosoft. 2011). Katso luku 3.3.1 järjestelmään kirjautumisesta.



Kuvio 8. Järjestelmän ulkoinen rakenne

Kuviossa 9 on kuvattu järjestelmän rakenne sen moduuleineen ja tietokantojen rakenteineen. Järjestelmässä on viisi moduulia, joita ovat tuotetietojen ylläpito, tuotetietojen vertailu, tuotedokumenttien ylläpito, järjestelmän ohjaustietojen ylläpito sekä moduuli järjestelmän toiminto ja virhetietojen eli ns. debug-tietojen seuraamiseen. Kolme ensimmäistä moduulia on tarkoitettu tuotetietotaulujen sisällön hallinnoimiseen. Kaksi viimeistä moduulia hallinnoivat tai näyttävät tietoja järjestelmän ohjaustauluista.



Kuvio 9. Järjestelmän rakennekaavio

Tuotetietojen ylläpidon kautta hallinnoidaan tietokantojen tuotetietotauluissa olevia tietoja. Nämä tiedot ovat tekstiä, lukuja ja kuvia, ja tämän moduulin kautta ei voida vaikuttaa lopullisen tuotedokumentin ulkoasuun. Tuotetietojen vertailumo-

duulin kautta tuotetietotaulujen sisältöä voidaan taulukoida eri hakuehdoilla peräkkäin samaan taulukkoon näkyväksi. Tämän näkymän avulla tiedot voidaan tarkistaa tai niitä voidaan vertailla nopeasti. Kolmas moduuli on tuotedokumenttien ylläpito ja sen kautta ylläpidetään tuotedokumenttien ulkoasua ja visuaalista rakennetta. Moduulin avulla tyhjälle tuotedokumenttipohjalle voidaan lisätä taulukoita, kuvia ja tietokantakenttiä niin sanottuina kirjanmerkkeinä eli bookmarkeina. Kirjanmerkit on sidottu eri tuotetietotaulujen kenttiin ja tyhjä tuotepohja ja sen tietokantakentät voidaan napista painamalla täyttää tietokannan tiedoilla. Neljännen moduulin kautta ylläpidetään järjestelmän ohjaustietoja ja muuta järjestelmään liittyvää informaatiota. Tämän kautta voidaan muun muassa muuttaa valikoiden sisältöä, hallinnoida käyttäjäoikeuksia ja muuttaa tuotedokumenttien muotoilusääntöjä. Viimeisenä moduulina on toimintonäkymä, jonka kautta voidaan tarkistaa viimeisimmän toiminnon tai sql-lauseen sisältö. Virhetilanteissa tämän näkymän kautta voidaan saada mahdollista tietoa siitä, mikä aiheutti järjestelmän virheellisen toiminnan.

Liitteinä 4 ja 5 on vielä tarkempi rakennekaavio tuotetieto- ja järjestelmätauluista sekä järjestelmän ohjaustauluista. Liitteistä löytyy taulukohtainen selostus taulujen sisältämistä kentistä ja niiden tietotyypeistä.

4.3.1 Järjestelmään kirjautuminen

TuokeSpecs-ohjelman käynnistyessä ensimmäisenä avautuu ikkuna, joka kertoo kirjautuneena olevan käyttäjän ja hänen käyttäjäryhmänsä järjestelmässä (Kuvio 10). Lisäksi näytetään myös käytössä olevan koneen nimi ja ip-osoite. Näiden tietojen alla on valintanapit, joiden avulla käyttäjä voi valita avattavan tietokannan. Tätä valintaa ei voi muuttaa muuten kuin avaamalla ohjelma uudestaan.

Kuvio 10. Järjestelmän kirjautumisikkuna

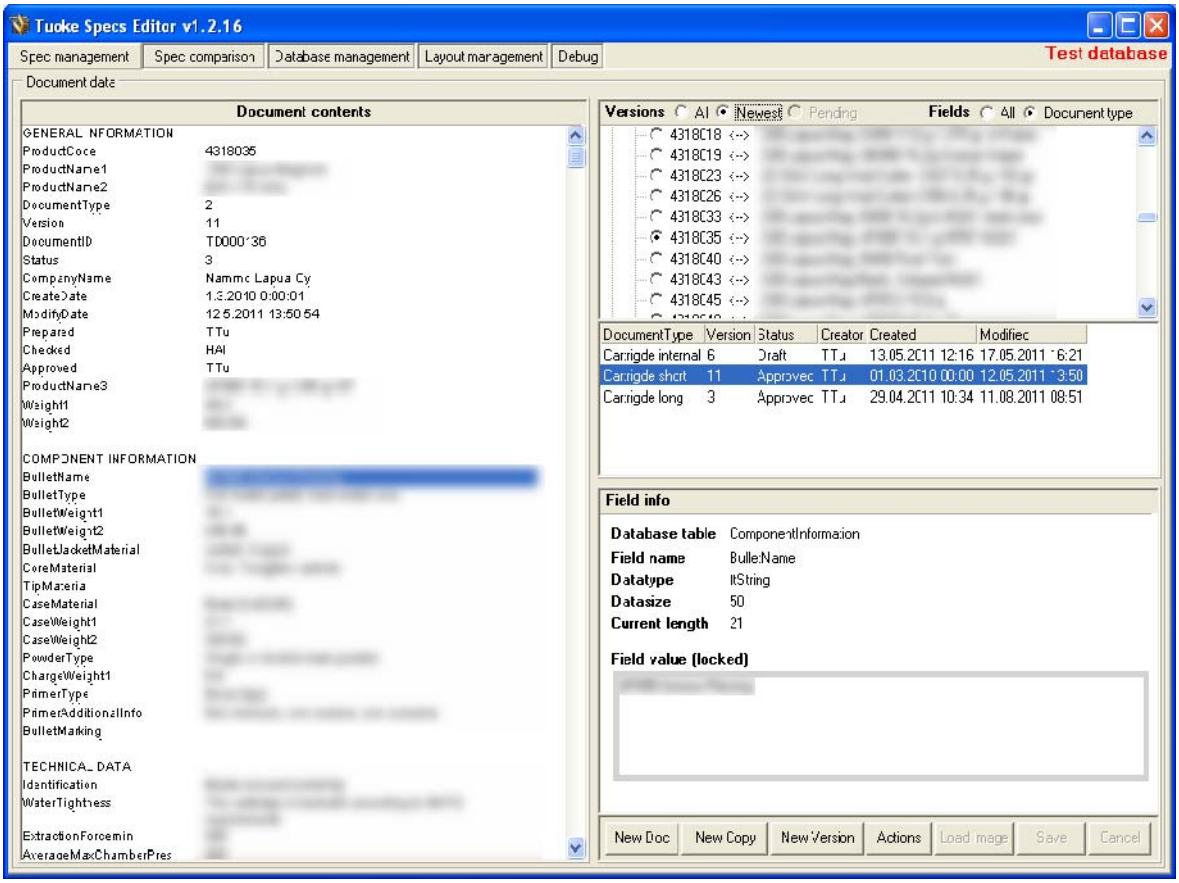
Aloitusikkuna ja kirjautuminen on toteutettu käyttämällä Windows API-ohjelmointirajapinnan valmiita funktioita, joiden avulla ohjelmakoodissa voidaan selvittää tällä hetkellä kirjautuneena olevan käyttäjän nimi, käytössä olevan koneen nimi ja ip-osoite. Toteutuksessa käytettiin netistä helposti löydettävissä olevia Windows API -dokumentteja ja niistä tehtyjä ohjelmointiesimerkkejä. Loput komponenteista ovat Delphin mukana tulevia vakiokomponentteja eli tekstikenttä, nappi ja kuva. Kyseisen toteutuksen avulla järjestelmä saatiin käyttäjienhallinnan puolesta integroitua Microsoft Active Directory-toimialueen käyttäjienhallintaan siltä osin, että käyttäjien ei tarvitse erikseen kirjautua järjestelmään. Lisäksi salasanat hallinnoidaan Windows-toimialueen käyttäjienhallinnan ylläpidon kautta normaalilla tavalla, kuten Windows-ympäristöstä on tuttua. Näin ollen TuokeSpecs-järjestelmän ei tarvitse missään vaiheessa tietää käyttäjän salasanaa, lisäksi salasanan vaihdos toimialueelle näkyy heti myös TuokeSpecs-järjestelmälle kirjautumisen yhteydessä.

Ohjelmakoodina toteutus on seuraavanlainen: muuttujiin userName, hostName ja ipAddress tallennetaan vastaavat tiedot myöhempää tarvetta varten ja samat tiedot tulostetaan myös kirjautumisikkunaan kuviossa 10 näkyvällä tavalla. Funktiot getLoginName ja getIpFromHost hyödyntävät ja implementoivat Windows API-ohjelmointirajapinnan omia funktioita, joilla tarvittavat tiedot voidaan hakea käytössä olevalta koneelta ja kirjautuneen käyttäjän tiedoista.

Esimerkki 1. Kirjautuminen käyttäen Windows API-funktioita

```
userName := getLoginName;
getIpFromHost(hostname,ipAddress,strError);
```

4.3.2 Tuotetietojen ylläpito: käyttöliittymä



Kuvio 11. Tuotetietojen ylläpito

TuokeSpecs-ohjelman ensimmäinen osa tai moduuli on tuotetietojen ylläpito eli Spec management. Se löytyy ohjelman ensimmäiseltä välilehdeltä. Tämän välilehden kautta ohjelman avulla voidaan ylläpitää tuotanto- tai testitietokantojen tuotetietotaulujen sisältöä. Tuotetietotaulujen rakenteen kuvaus löytyy kuvoista 9 ja liitteistä 4 ja 5.

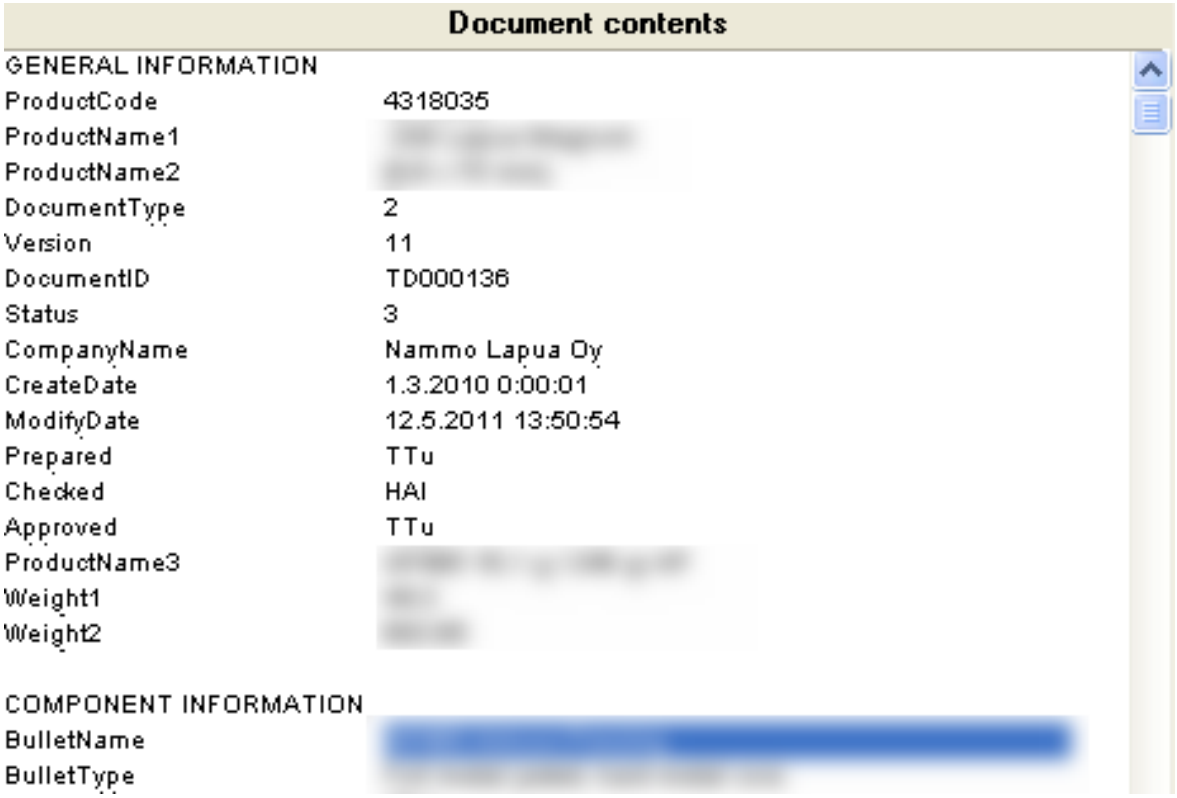
Moduuli on rakennettu Delphin vakio-komponenteilla, tekstikentillä, paneeleilla, listoilla, napeilla, radionapeilla ja puunäkymällä.

Kuvio 12. Tuoteluettelo

Kuviossa 11 on oikeassa reunassa ylhäällä tietokannan tietojen mukaan täytetty luettelo tämän hetkisistä tuotteista ja niiden kuvaukset. Sama luettelo on kuviossa 12 isompana. Luettelon yhteydessä ylempänä ovat kentät, joiden avulla tietokannasta tuotavia tietoja voidaan rajata. Vasemmalla on valinta, jolla dokumenttilistan dokumentit voidaan rajoittaa vain uusimpiin versioihin tai vaihtoehtoisesti tuodaan näkyville kaikki haettavana olevaan tuotteeseen liittyvät tuotedokumentit. Oikealla on valinta, jonka avulla rajataan avatun dokumentin kenttien määrää. Näkyville voidaan tuoda joko kaikki kentät tai tietokantaan ennalta määritellyt tietyn tuotedokumentin useimmin käytettävät kentät.

DocumentType	Version	Status	Creator	Created	Modified
Cartrigde internal	6	Draft	TTu	13.05.2011 12:16	17.05.2011 16:21
Cartrigde short	11	Approved	TTu	01.03.2010 00:00	12.05.2011 13:50
Cartrigde long	3	Approved	TTu	29.04.2011 10:34	11.08.2011 08:51

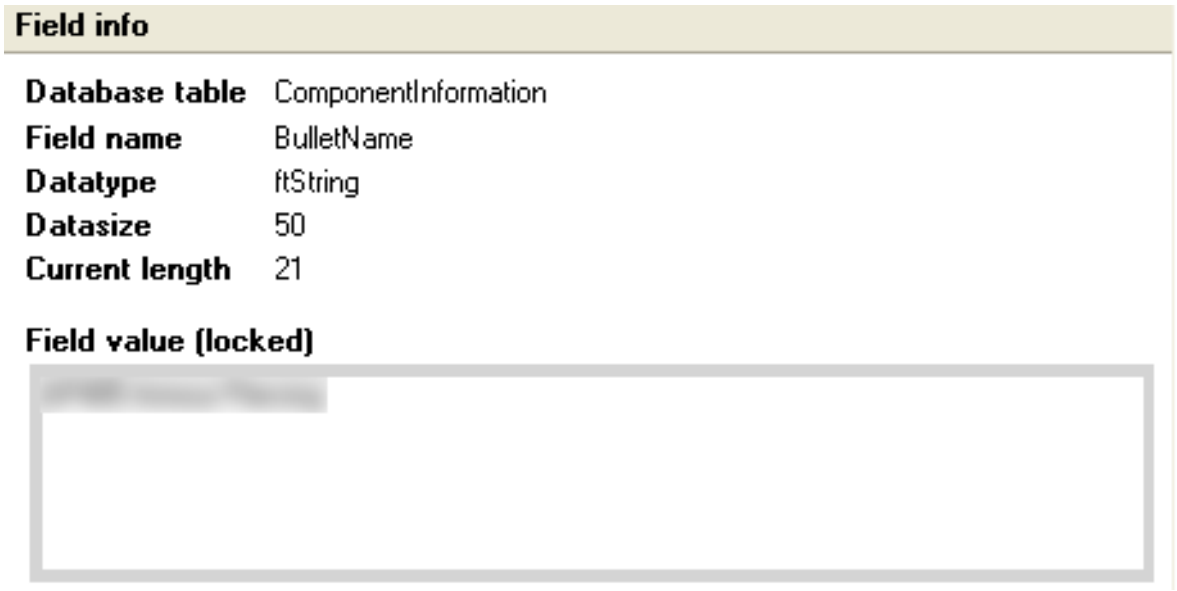
Kuvio 13. Tuotedokumenttiluettelo



Kuvio 14. Tuotedokumentin sisältämät tiedot

Valittuna olevaan tuotteeseen liittyvät dokumentit tulevat näkyviin omaksi listakseen tuotelistan alle, kuten kuviosta 13 näkyy. Dokumenteista tuodaan näkyviin dokumentin tyyppi, versio, tämän hetkinen status, dokumentin luoja sekä luonti ja muokkauspäivät. Tästä listasta käyttäjä voi valita edellä mainittujen tietojen perusteella avattavan dokumentin. Listasta valitun dokumentin tiedot avautuvat ikkunan vasempaan reunaan. Ensin näytetään kentän nimi ja sitten sen sisältämä tieto.

Valitsemalla kentän tiedot kuvion 14 kenttälistauksesta, niitä pääsee muokkaamaan ikkunan alareunassa olevaan näkymään, joka on esitetty kuviossa 15. Tämän näkymän kautta tietokannan eri kenttiin voidaan syöttää niin tekstiä, numeroita, kuvia ja ennalta määritellyn muotoisia tekstitiedostoja.



Kuvio 15. Tuotetietojen muokkaus

4.3.3 Tuotetietojen ylläpito: tietojen haku

Ohjelmakoodissa dokumentin avaus on toteutettu seuraavasti aliohjelmassa haeDokumentti, joka suoritetaan, kun dokumenttilistan dokumenttia kaksoisnapsautetaan: aluksi luodaan dynaamisesti TADOQuery-objekti, adoDocFields, jolle annetaan parametreiksi dokumentin avain-kentät, tuotekoodi, dokumenttityyppi ja versio. Sen jälkeen kutsutaan aliohjelmaa fillGrid vuorollaan joka tuotetietotaululle, eli GeneralInformation, ComponentInformation, TechnicalData, PackingAndMarking, QualityInspection, ReferencedTechicalData ja Drawings. FillGrid-aliohjelma täyttää tuotedokumentin tiedot kustakin tietokannan taulusta vasemmassa reunassa olevaan taulukko-komponenttiin. Lisäksi haetaan tuotedokumenttiin mahdollisesti liittyvät kuvat Drawings-taulusta ja mahdollinen ballistinen ampumadata TechnicalData -taulusta TMemoryStream-objekteihin eli muistivirtoihin. Tarkoituksena on hakea kaikki dokumenttiin liittyvä tieto kerralla ohjelman muistiin, jotta dokumenttia editoidessa ei tarvitse koko ajan hakea tietoa tietokannasta.

Esimerkki 2. Tuotetietojen haku

```
procedure Tlomake1.haeDocumentti(sqlPCode : string; sqlDocType : string; sqlVerNumber :
string; saveEv : boolean);
var
  qDrawings : TAdoQuery;
  adoDocFields : TADODataset;
  ds : TDataset;
  iTemp : integer;
  mode : String;
begin
    ......
    adoDocFields := TADODataset.Create(nil);
    adoDocFields.Connection := connSpecmanage;
    adoDocFields.CommandText := 'getDocFields';
    adoDocFields.CommandType := cmdStoredProc;
    adoDocFields.Parameters.AddParameter;
    adoDocFields.Parameters[0].Name := 'TableName';
    adoDocFields.Parameters[0].DataType := ftString;
    adoDocFields.Parameters[0].Value := 'GeneralInformation';
    ....
    adoDocFields.Open;
    OpenSpecDoc := getDocKeyinformation(sqlPCode,strtoint(sqlDocType),
              strtoint(sqlVerNumber));
    fillGrid(adoDocFields,Stringgrid1,false,'General Information','GeneralInformation',true);
    adoDocFields.Close;
    adoDocFields.Parameters[0].Value := 'ComponentInformation';
    adoDocFields.Open;
    fillGrid(adoDocFields,Stringgrid1,true,'Component Info','ComponentInformation',false);
    adoDocFields.Close;
    adoDocFields.Parameters[0].Value := 'TechnicalData';
    adoDocFields.Open;
    fillGrid(adoDocFields,Stringgrid1,true,'Technical Data','TechnicalData',false);
    adoDocFields.Close;
    adoDocFields.Parameters[0].Value := 'PackingAndMarking';
    adoDocFields.Open;
    fillGrid(adoDocFields,Stringgrid1,true,'Packing And Marking','PackingAndMarking',false);
    adoDocFields.Close;
    ....

    qDrawings := TAdoQuery.Create(Self);
    qDrawings.Connection := currConnection;
    qDrawings.SQL.Text :=
```

```
    'SELECT * FROM Drawings WHERE ProductCode='''+sqlPCode+
    ''' AND DocumentType='+sqlDocType+ ' AND Version='+sqlVerNumber;
  qDrawings.Active := true;

  if qDrawings.RecordCount > 0 then
  begin
    if not qDrawings.FieldByName('CompDrawing').IsNull then
    begin
      load-
BlobImage(specImage1,qDrawings.FieldByName('CompDrawing'),CompDrawingStream);
    end;
    if not qDrawings.FieldByName('BulletPath').IsNull then
    begin
      loadBlobImage(specImage2,qDrawings.FieldByName('BulletPath'),BulletPathStream);
    end;
    if not qDrawings.FieldByName('BulletVelocity').IsNull then
    begin
      load-
BlobImage(specImage3,qDrawings.FieldByName('BulletVelocity'),BulletVelocityStream);
    end;
  end;

  ....
  qDrawings.SQL.Text :=
    'SELECT BallisticData FROM TechnicalData WHERE ProductCode='''+sqlPCode+
    ''' AND DocumentType='+sqlDocType+' AND Version='+sqlVerNumber;

  qDrawings.active := true;
  if qDrawings.RecordCount > 0 then
  begin
    if not qDrawings.FieldByName('BallisticData').IsNull then
    begin
      memo1.Lines.Text := qDrawings.FieldByName('BallisticData').AsString;
      BallisticDataStream.Clear;
      memo1.Lines.SaveToStream(BallisticDataStream);
      memo1.Lines.Clear;
    end;
  end;

  qDrawings.Active := false;
  qDrawings.Free;
end;
```

4.3.4 Tuotetietojen ylläpito: tietojen tallennus

Tuotetietojen tallennus on toteutettu ohjelmakoodissa seuraavasti aliohjelmassa tallennaDokumentti. Tarkoituksena oli rakentaa Microsoft Wordin kaltainen ja tekstinkäsittelystä muutenkin tuttu tallennustapa eli kaikki tallennetaan kerralla yhdellä napin painalluksella, eikä yksittäisiä muutoksia käydä heti siirtämässä tietokannan tauluihin. Aluksi tutkitaan löytyykö avainkentillä haettaessa jo tuotedokumentin tietoja taulusta GeneralInformation. Jos tarkoituksena on luoda uusi dokumentti ja avaintiedoilla löytyy jo tietokantarivejä, annetaan käyttäjälle varoitus tilanteesta ja peruutetaan tietojen tallennus. Jos dokumentti ei ole uusi tai tietoja ei löydy, jatketaan tallennusta ja käytetään edellisen kyselyn tulosta määrittämään luodaanko tietojen tallennusta varten Insert- vai Update-lause.

Sql-lauseen muodostamista varten StringGrid-komponentti, jossa tuotedokumentin tiedot ovat, käydään läpi alusta loppuun silmukassa, for i := 0 to Grid.RowCount-1 do. Jokaisen taulukkorivin kohdalla tutkitaan onko tietoja muutettu ja, että rivi ei ole ns. tietokannan laskettu kenttä, joiden tietoja ei voi päivittää update- tai insert-lauseella. Tämä tehdään if-ehdolla, if (Grid.Cells[8,i] = 'C') and (Grid.Cells[7,i] <> 'cal'). Kun Sql-lause on muodostettu, suoritetaan se ja tallennetaan sen avulla muuttuneet tiedot tietokannan eri tauluihin.

Lopuksi tarkistetaan vielä onko tuotetietojen kuvia tai ballistista dataa muutettu tietojen muokkauksen aikana. Ne tarkistetaan seuraavien if-ehtojen avulla, if (PictureChanged[0] = true) and (CompDrawingStream.Size > 0) ja if BallisticDataStream.Size > 0. Jos kuvia tai ballistista dataa on muutettu, muuttuneet tiedot tallennetaan tietokannan kenttiin muiden tuotetietojen kanssa.

Esimerkki 3. Tuotetietojen tallennus

```
procedure Tlomake1.tallennaDocumentti(newDoc : boolean; Grid : TStringGrid; saveEv : boole-
an);
var
 i : integer; BlobField: TBlobField; Taulut : TStringList;
 commEventLog, commBallisticData : TAdoCommand;
 queryDrawings, queryTauluCheck : TAdoQuery;
 sqlParam, par : TParameter;
 ...
begin
 ...
 Taulut := TStringList.Create;
 adoquery3.SQL.Text := 'SELECT * FROM GeneralInformation WHERE ProductCode=...';
 adoquery3.Open;

 if newDoc and (adoquery3.RecordCount > 0) then
 begin
   docFound := true;
   showmessage('Document found with keyvalues :'+..... 'Data not saved.');
 end;

 if newDoc and not docFound then
 begin
    strTaulu := 'GeneralInformation'; strSQL := 'INSERT INTO GeneralInformation (';
    commEventLog.Parameters[3].Value := 'db-insert (document)';

  for i := 0 to Grid.RowCount-1 do
  begin
    if (Grid.Cells[5,i] <> '') and (Grid.Cells[7,i] <> 'cal') then
    begin
      if strTaulu = Grid.Cells[4,i] then
      begin
        if strFields = '' then strFields := strFields + Grid.Cells[0,i]
        else strFields := strFields +','+ Grid.Cells[0,i];

        if strValues = '' then
        begin
          strValues := strValues +teeSQLArvo(Grid.Cells[1,i],Grid.Cells[2,i]);
        end
        else
        begin
          strValues := strValues +','+ teeSQLArvo(Grid.Cells[1,i],Grid.Cells[2,i]);
```

```
        end;
       ....
      end
      else
      begin
       strTaulu := Grid.Cells[4,i];
       if keyFields then
       begin
         strSQL := strSQL+strFields+') VALUES ('''+
         sqlProductCode+''','+sqlDocumentType+','+inttostr(sqlVersion)+','+strValues+')';
       end
       else
       begin
         strSQL := strSQL+strFields+') VALUES ('+strValues+')';
       end;
       adoquery3.SQL.Add(strSQL);
       strSQL := 'INSERT INTO '+strTaulu+' (ProductCode,DocumentType,Version,';
       strFields := Grid.Cells[0,i];
       strValues := teeSQLArvo(Grid.Cells[1,i],Grid.Cells[2,i]);
      end;
    end;
  end;

  strSQL := strSQL+strFields+') VALUES ('''+
  sqlProductCode+''','+sqlDocumentType+','+inttostr(sqlVersion)+','+strValues+')';
end
else
begin
  commEventLog.Parameters[3].Value := 'db-update (document)';
  if sqlDocumentType = '1' then newVersions := true;

  for i := 0 to Grid.RowCount-1 do
  begin
    if (Grid.Cells[8,i] = 'C') and (Grid.Cells[7,i] <> 'cal') then
    begin
      if (strTaulu = '') then
      begin
        strTaulu := Grid.Cells[4,i]; strSQLStart := 'UPDATE '+strTaulu+' SET ';
        Taulut.Add(strTaulu);
      end;

      fieldsUpdated := true;
      if strTaulu = Grid.Cells[4,i] then
```

```
    begin
     if strFields = '' then
     begin
      strFields := strFields + Grid.Cells[0,i] + '='+teeSQLArvo(Grid.Cells[1,i],Grid.Cells[2,i]);
      eventTemp := Grid.Cells[0,i];
     end
     else
     begin
      strFields:=strFields +','+ Grid.Cells[0,i] + '='+teeSQLArvo(Grid.Cells[1,i],Grid.Cells[2,i]);
      eventTemp := eventTemp+';'+Grid.Cells[0,i];
     end;
    end
    else
    begin
     if eventDetails1 <> '' then
     ...
     else
     ...
     strTaulu := Grid.Cells[4,i];
     if (strFields <> '') then
     begin
        strSQL := strSQLStart+strFields+ ' WHERE ProductCode=.... ';
      adoquery3.SQL.Add(strSQL);
     end;
     strSQLStart := 'UPDATE '+strTaulu+' SET ';
     Taulut.Add(strTaulu);
     ....
    end;
  end;
end;

if (strFields <> '') then
begin
 strSQLStart := strSQLStart + strFields;
 strSQL := strSQLStart +' WHERE ProductCode='''+sqlProductCode+
  ''' AND DocumentType='+sqlDocumentType+' AND Version='+inttostr(sqlVersion);

 if pos('=',eventDetails1) > 0 then
 begin
   eventDetails1 := eventDetails1+'@'+strTaulu+'='+eventTemp;
 end
 else
 begin
```

```
      eventDetails1 := strTaulu+'='+eventTemp;
    end;
  end
  else
    strSQLStart := '';
end;
adoquery3.Close;

if not docFound then
begin
  if OpenSpecDoc.DocModified then
  begin
    saveDocKeyinformation(OpenSpecDoc);
  end;

  if (strSQL <> '') then
  begin
    try
      ...
      for i := Taulut.Count -1 downto 0 do
      begin
        queryTauluCheck.SQL.Text :=
        'SELECT * FROM ' + Taulut.Strings[i] +' WHERE ProductCode='''+sqlProductCode+
        ''' AND DocumentType='+sqlDocumentType+ ' AND Version='+inttostr(sqlVersion);

        queryTauluCheck.Open;
        if queryTauluCheck.RecordCount = 1 then
        begin
          Taulut.Delete(i);
        end;
        queryTauluCheck.Close;
      end;

      if (Taulut.Count > 0) then
      begin
        for i := 0 to Taulut.Count -1 do
          queryTauluCheck.SQL.Add('INSERT INTO ' + Taulut[i] +
            ' (ProductCode,DocumentType,Version) VALUES ('''+
            sqlProductCode+''','+sqlDocumentType+','+inttostr(sqlVersion)+')');
        queryTauluCheck.ExecSQL;
      end;
      queryTauluCheck.Free;
      adoQuery3.SQL.Add(strSQL);
```

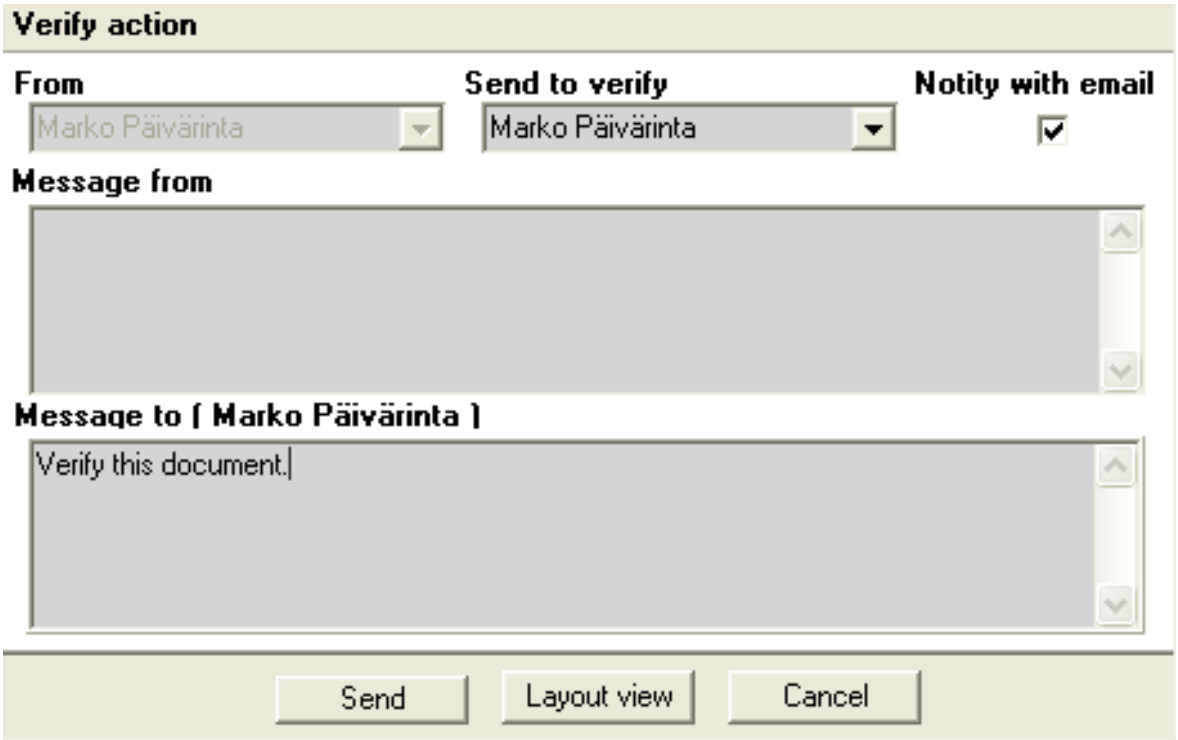
```
      adoQuery3.ExecSQL;
      ...
    except
      makeDebugInfo(adoQuery3.SQL.text);
      showmessage('SQL Error, last action cancelled!');
    end;
  end;

  if (PictureChanged[0] = true) or .....  then
  begin
    ...
    queryDrawings.SQL.Text := 'SELECT * FROM Drawings WHERE ProductCode=....';
    queryDrawings.Open;
    if queryDrawings.RecordCount = 0 then
    begin
      queryDrawings.SQL.Text :=  'INSERT INTO Drawings '+...')';
      queryDrawings.ExecSQL;
    end
    else
      queryDrawings.SQL.Text := 'SELECT * FROM Drawings WHERE.... ';
       ...

    if (PictureChanged[0] = true) and (CompDrawingStream.Size > 0) then
    begin
      CompDrawingStream.Position := 0;
      TBlobField(
        queryDrawings.FieldByName('CompDrawing')).LoadFromStream(...);
    end;
    ...
    queryDrawings.Post; queryDrawings.Active := false: queryDrawings.Free;
  end;
  if BallisticDataStream.Size > 0 then
  begin
    ...
  end;
  ...
  / / Column 8 is the Field-change marker, reseting it.
  for i := 0 to StringGrid1.RowCount -1 do StringGrid1.Cells[8,i] := '';
 end;
end;
```

4.3.5 Tuotetietojen hyväksymisrutiini

Järjestelmä mahdollistaa kolmivaiheisen tuotedokumenttien hyväksyntämenettelyn. Kaaviokuva siitä löytyy liitteestä 2. Hyväksymismenettely aloitetaan avaamalla draft-tilassa oleva dokumentti. Tämän jälkeen se voidaan lähettää vahvistettavaksi dokumenttinäkymän alareunassa olevien nappien kautta.

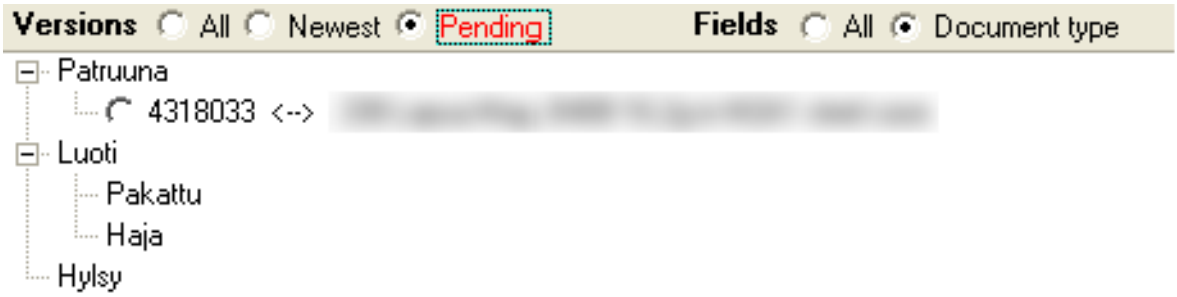
Kuvio 16. Hyväksyntämenettelyn aloitus

Kun edellinen toimenpide on tehty, avautuu ikkunan oikeaan alareunaan näkymä, jonka avulla valitaan kenelle dokumentti lähetetään vahvistettavaksi. Tässä näkymässä voidaan myös kirjoittaa kyseiselle henkilölle viesti ja valita saako hän ilmoituksen tapahtumasta sähköpostiinsa.



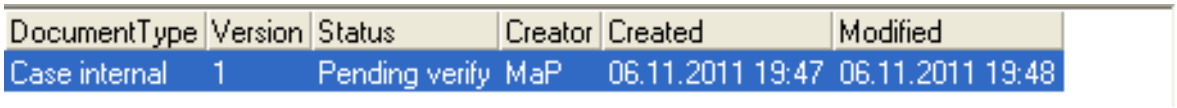
Kuvio 17. Dokumentin lähetys vahvistettavaksi

Kun dokumentti on lähetetty vahvistettavaksi toiselle käyttäjälle, tämä kyseinen käyttäjä näkee kirjautuessaan tuotehallintamoduulin yläreunassa ilmoituksen siitä, että hänellä on odottamassa vahvistettavia dokumentteja. Tämän pikavalinnan avulla käyttäjä näkee suoraan toimenpiteitä odottavat dokumentit ilman, että niitä pitää etsiä itse puunäkymän sisällöstä.



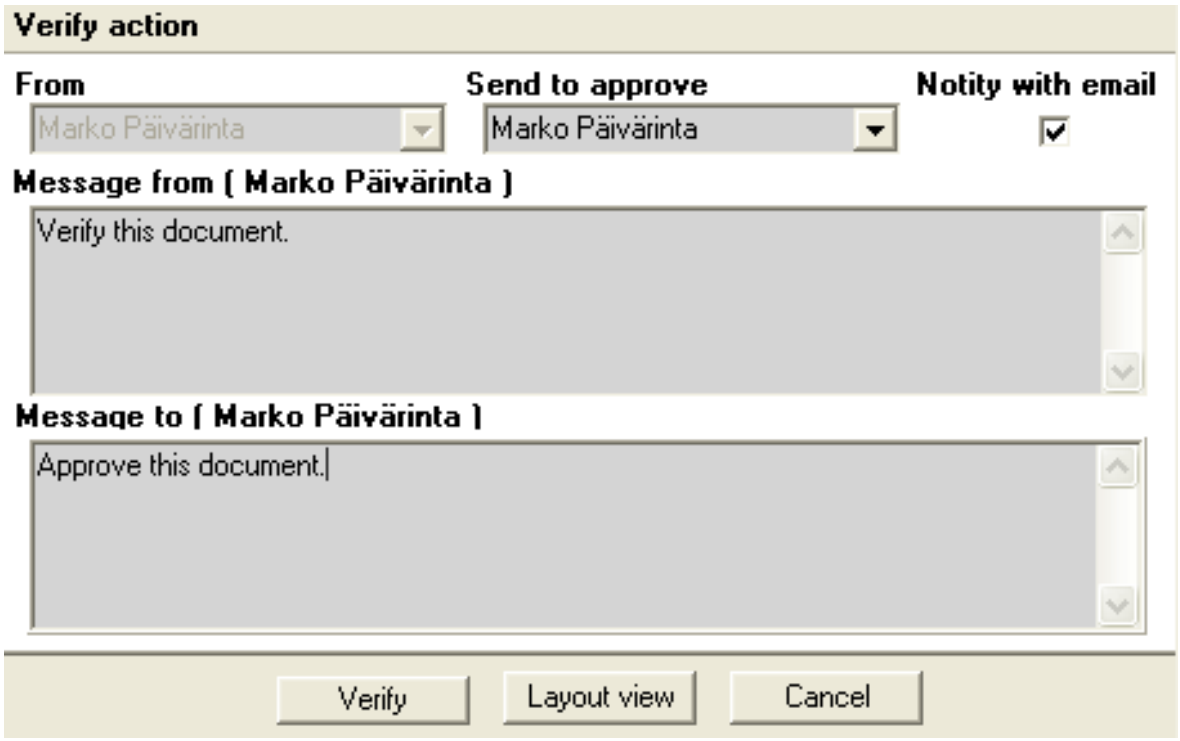
Kuvio 18. Vahvistettavan dokumentin pikasuodatus

Puunäkymästä valittuaan käyttäjä näkee tuotekoodilla olevat dokumentit alempana listassa. Tämän listan kautta näkyy myös dokumentin tila, joka on tässä tapauksessa Pending verify.



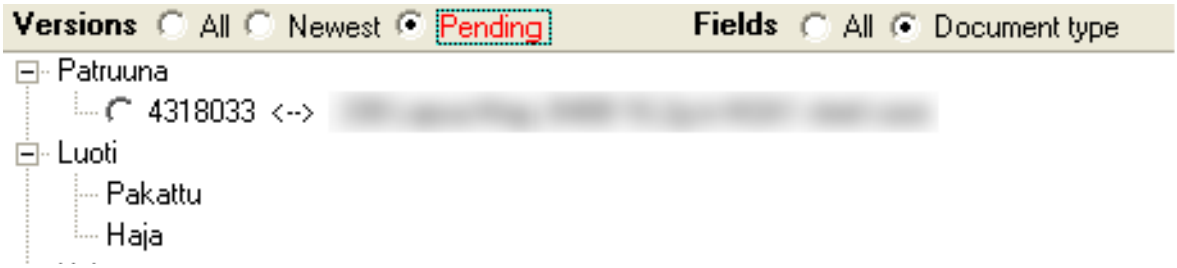
Kuvio 19. Dokumentti odottaa vahvistusta

Avattuaan dokumentin listasta kaksoisnapsauttamalla ruudun alareunaan avautuu taas samanlainen näkymä kuin aikaisemmin, mutta nyt käyttäjä voi näkymän avulla vahvistaa kyseisen dokumentin ja lähettää sen valitsemalleen käyttäjälle hyväksyttäväksi. Tässä näkymässä näkyy mahdollinen viesti, jonka edellinen käyttäjä on lähettänyt dokumentin vahvistajalle. Lisäksi näkymässä voi jälleen kirjoittaa viestin henkilölle, jolle dokumentti lähetetään vahvistettavaksi, sekä muistuttaa käyttäjää sähköpostilla asiasta. Käyttäjän painaessa nappia ja lähettäessä dokumentin vahvistettavaksi järjestelmä päivittää käyttäjän kolmikirjain tunnuksen tietokannan GeneralInformation-taulun Checked-kenttään. Näin dokumentti on tarkistettu ja se on valmis hyväksyttäväksi.



Kuvio 20. Dokumentin lähetys hyväksyttäväksi

Käyttäjä, jolle dokumentti lähetetään vahvistettavaksi, näkee jälleen aukaistessaan järjestelmän valintanapin, jolla saa puunäkymään vain toimenpiteitä odottavat dokumentit.



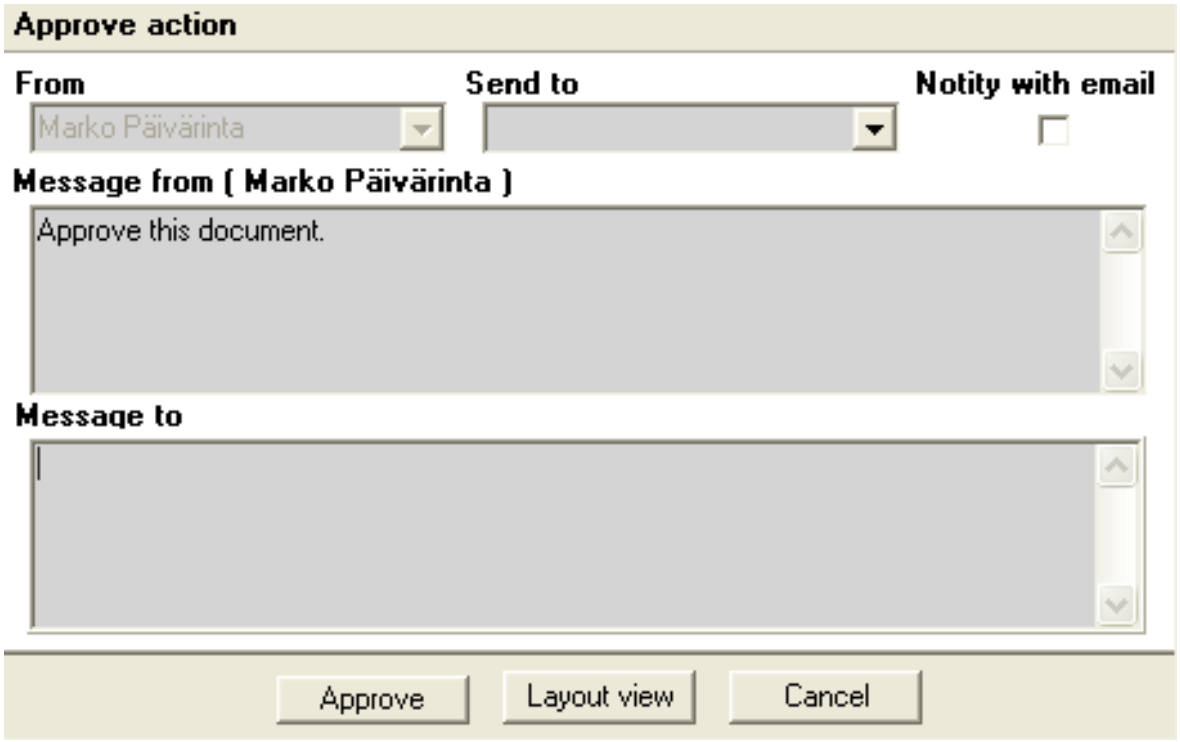
Kuvio 21. Hyväksyttävän dokumentin pikasuodatus

Valittuaan puunäkymästä tuotekoodin alempana olevaan listaan avautuu tuotedokumentti, jonka tila on Pending approve.

DocumentType	Version	Status	Creator	Created	Modified
Case internal	1	Pending approve	MaP	06.11.2011 19:47	06.11.2011 19:48

Kuvio 22. Dokumentti odottaa hyväksyntää

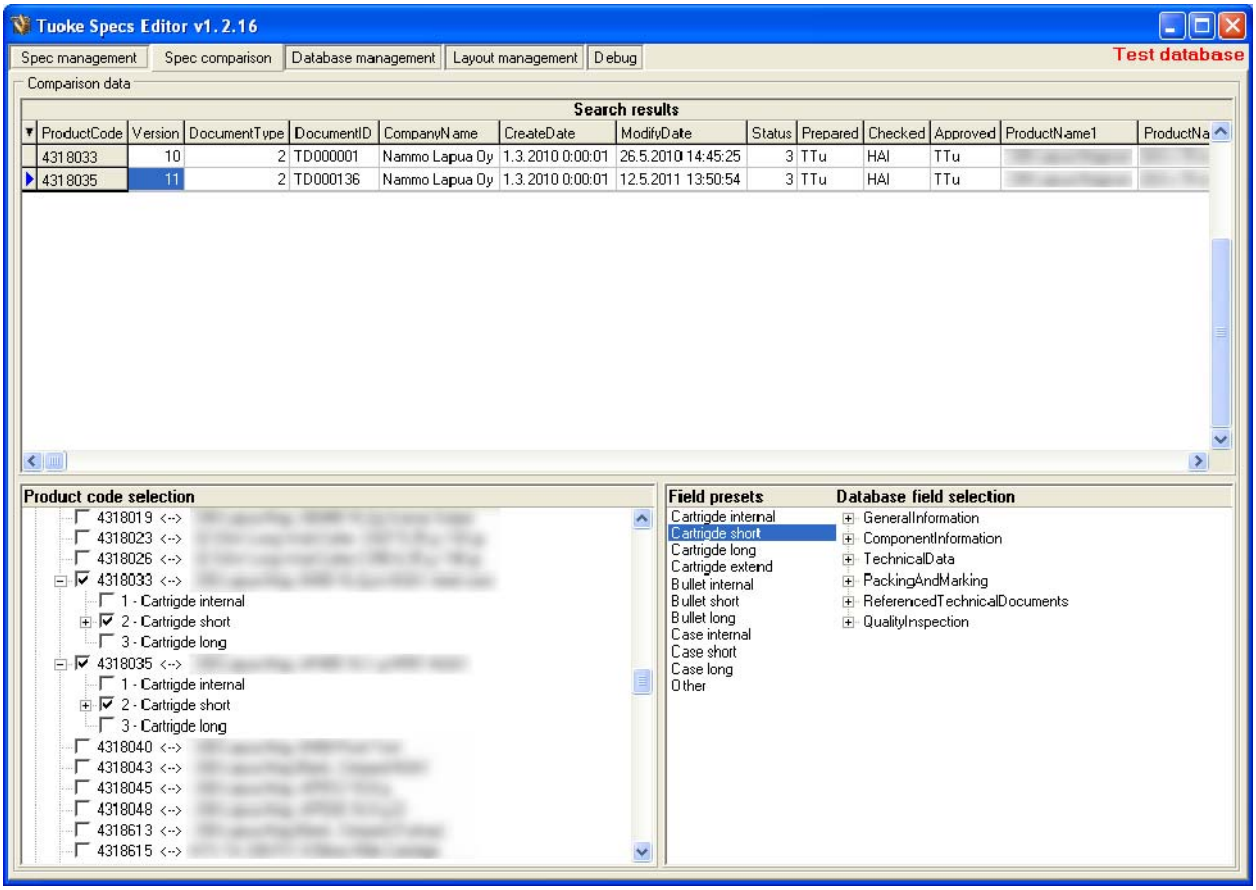
Dokumentin avaamisen jälkeen käyttäjä näkee ruudun vasemmassa näkymän, jolla voidaan vahvistaa dokumentti. Vahvistuksen jälkeen sen tila tietokannassa muuttuu Approvediksi. Näkymän kautta on mahdollista lähettää vielä myös viesti toiselle käyttäjälle, mutta dokumentin hyväksymiskierros on tämän toimenpiteen jälkeen valmis eli mahdollinen viesti on vain informatiivinen tieto. Tuotedokumentti on hyväksymisen jälkeen lukittu, eikä sen tietoja voi enää muuttaa ilman, että dokumentista luodaan joko uusi kopio tai versio.



Kuvio 23. Dokumentin vahvistus

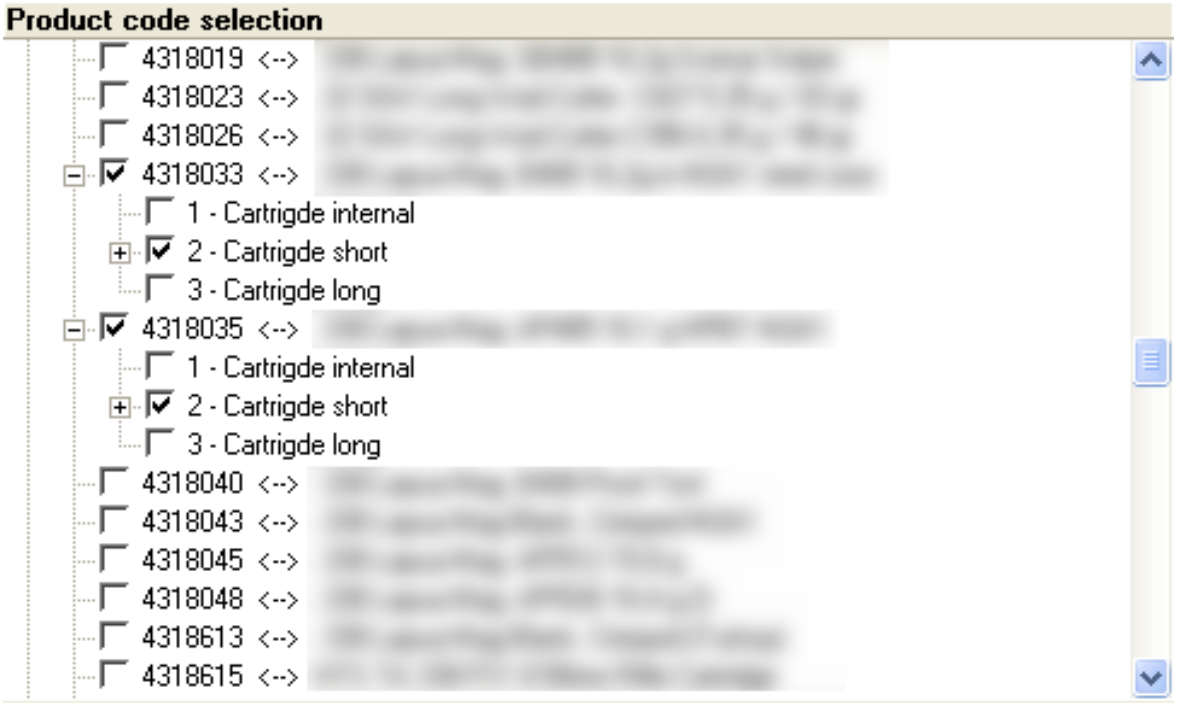
4.3.6 Tuotetietojen vertailu

Ohjelman toinen moduuli on tuotetietojen vertailu. Tämän näkymän kautta tietokannassa olevia tietoja voidaan vertailla taulukoimalla niitä eri hakuehtojen avulla näkyville.



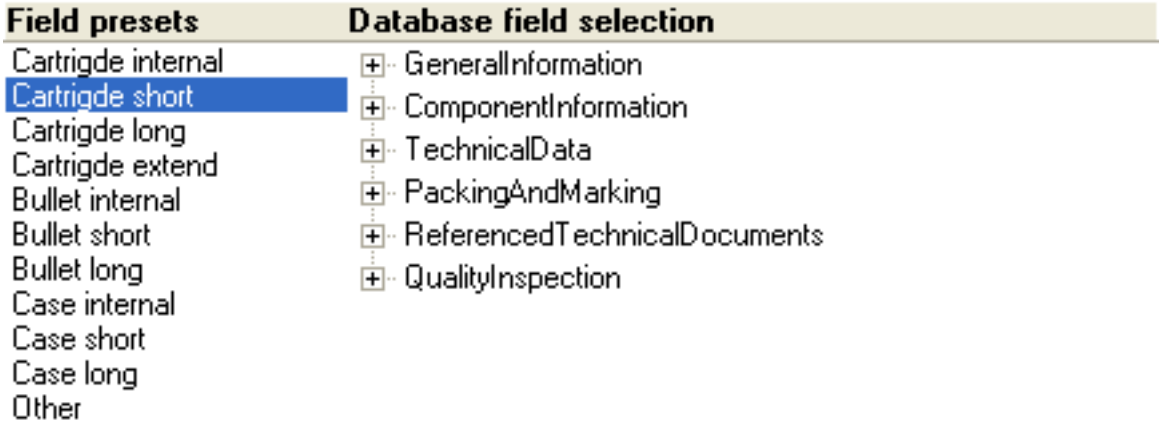
Kuvio 24. Tuotetietojen vertailu

Ruudun vasemmassa alareunassa on näkymä, josta voi valita kaksi tai useampia vertailtavia tuotteita. Valittujen tuotteiden tiedot listataan ylempänä olevaan näkymään. Puunäkymän avulla tuotteista voi valita vertailtavaksi eri tuotedokumenttityyppejä ja dokumenttien eri versioita. Näkymä on toteutettu Delphin vakiokomponenteilla, vain taulukkonäkymää varten on haettu ilmaiseksi haettavissa oleva paranneltu versio vakiotaulukosta. Muuten käytössä on Delphistä vakiona löytyvät tekstikenttä ja puunäkymä. Oikean puolen puunäkymään viereen on yhdistetty samalla taustavärillä lista-komponentti.



Kuvio 25. Vertailtavien dokumenttien valinta

Oikealta löytyy työkalut näytettävien kenttien valintaan (kuvio 25). Vasemmalla on ennalta määritellyt esivalinnat eri dokumenttityypeille ja yksittäisiä kenttiä voi vielä valita tai poistaa oikealla olevan puunäkymän kautta.



Kuvio 26. Vertailtavien tietokantakenttien valinta

Ohjelmakoodissa vertailunäkymä on toteutettu seuraavasti. Käytettävän Sql-lauseen muodostusta varten on rakennettu aliohjelma teeSQLHakulause. Parametreinä välitetään puunäkymä, jossa jonka kautta hakukentät on valittu, PuuSELECT, ja filterWHERE-taulukko, joka sisältää sql-lauseen where-osan hakuehdot.

Hakuehtoja varten on järjestelmään määritelty luokat, DocumentTypeFilter, sqlFilterItem ja sqlFilterList, SqlFilterItem on yksittäinen hakuehto ja sqlFilterList on yksittäisten hakuehtojen taulukko, jonka tiedot päivitetään aina, kun vertailumoduulin puunäkymän valintoja muutetaan. DocumentTypeFilter pitää sisällään kaikki yhdelle dokumenttityypille määritellyt hakuehdot.

Esimerkki 4. Tuotetietojen vertailu

```
type DocumentTypeFilter = record
  DocumentType : integer;
  DocDesc : String;
  NewestVersion : boolean;
  Versions : array of integer;
end;


type sqlFilterList = record
    index : integer;
    ProductCode : String;
    AllDocs : Boolean;
    DocumentTypes : array of DocumentTypeFilter;
end;


type sqlFilterList = class
  FilterItems : array of sqlFilterItem;
  procedure addFilterItem(ProductCode:String);
  procedure addDocType(ProductCode:string; DocumentType:integer);
  procedure deleteDocType(ProductCode:string; DocumentType : integer);
  procedure addVersion(ProductCode:String;
                              DocumentType:integer; Version:integer);
  procedure deleteVersion(ProductCode : String; DocumentType : integer; Version :
                                  integer);
  procedure deleteFilterItem(index : integer); overload;
  procedure deleteFilterItem(ProductCode : String); overload;
end;


procedure teeSQLHakulause(PuuSELECT : TTreeview; filterWHERE : sqlFilterList);
var
  strSQL, strTaulu, strFields : string;
  strArryTaulut : array of String;
  fieldsSelected : boolean;
```

```
  i, j, k : integer;
begin
.....
 for i := 1 to PuuSELECT.Items.Count -1 do
 begin
    ......
 end;
 ......
 for i := 0 to length(filterWHERE.filterItems) -1 do
 begin
   .....
 end;
end;
```

Aliohjelma toimii seuraavasti. Saatuaan parametreinään objektiviittauksen puunäkymään ja hakuehtotaulukkoon, se alkaa käydä niitä läpi kahdessa eri silmukassa. Ensimmäisessä vaiheessa käydään läpi ruudulla oikealla puolella oleva puunäkymä eli PuuSELECT, jossa on valittuna näkyville halutut kentät. Tästä puusta rakennetaan Sql-lauseen select-osan jälkeiset kentät, ja se on myös nimetty sen mukaan. Tämän silmukan aikana tutkitaan myös onko kenttiä valittuna ja tallennetaan sen mukainen arvo boolean-muuttujaan fieldsSelected. Myöhemmin tästä muuttujasta voidaan tarkistaa nopeasti, oliko yhtään kenttää valittu.

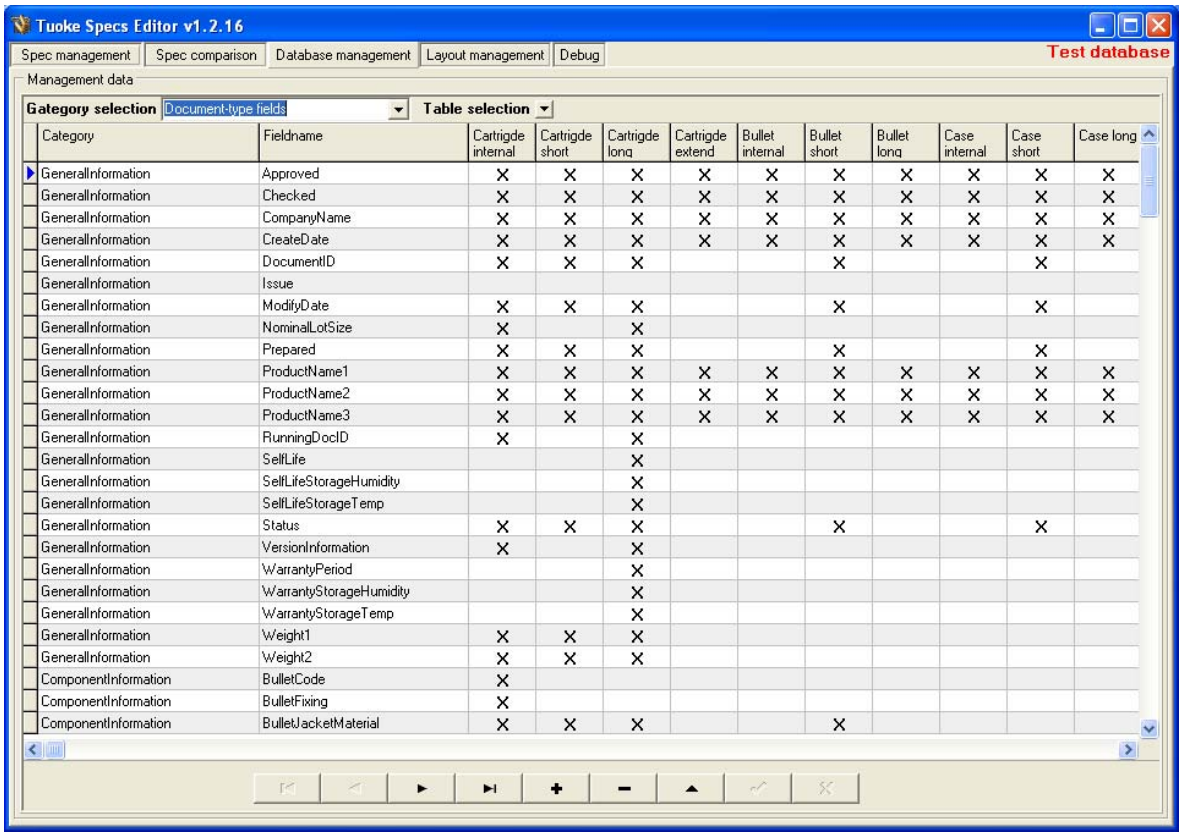
Toisessa silmukassa käydään läpi parametreinä saatu hakuehto-taulukko. Tämän taulukon tiedoista rakennetaan Sql-lauseen where-osa, ja se on myös nimetty sen mukaan. Esimerkinnä ohjelmassa silmukoilla rakennetun sql-lauseen sisältö:

Esimerkki 5. Sql-lause tuotetietojen vertailunäkymää varten

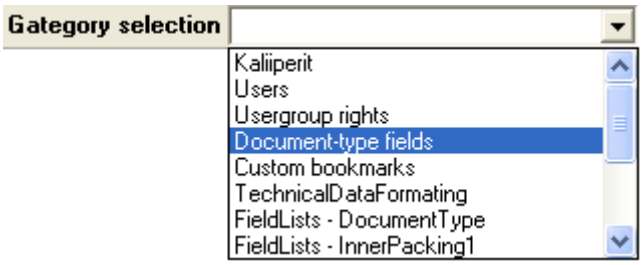
```
SELECT Gen.ProductCode, Gen.Version, Gen.DocumentType, Gen.CompanyName,
Gen.CreateDate, Gen.ModifyDate, Gen.Prepared, Gen.Checked, Gen.Approved,
Gen.ProductName1
FROM GeneralInformation Gen
WHERE (((Gen.ProductCode = '4318033') AND (((Gen.DocumentType = 2)  AND Gen.Version =
(SELECT Max(Version) FROM GeneralInformation Gen2 WHERE Gen.ProductCode =
Gen2.ProductCode AND Gen.DocumentType = Gen2.DocumentType)))) OR ((Gen.ProductCode =
'4318035') AND (((Gen.DocumentType = 2)  AND Gen.Version = (SELECT Max(Version) FROM
GeneralInformation Gen2 WHERE Gen.ProductCode = Gen2.ProductCode AND
Gen.DocumentType = Gen2.DocumentType)))))
```

4.3.7 Järjestelmän ohjaustietojen ylläpito

Ohjelman kolmas moduuli on järjestelmätaulujen hallinta. Järjestelmässä on useita tauluja, joilla ohjataan järjestelmän toimintaa. Näiden avulla muotoillaan muun muassa erilaisia tuotedokumenteille tulevia tauluja ja niiden tietoja sekä erilaisten tietokantakenttien yhteydessä listoja. Myös käyttäjäoikeuksien hallinta ja luvussa 4.3.2 kuvattua dokumenttien kenttien rajausta hallitaan näiden hallintamoduulin kautta. Käyttäjä voi valita ylläpidettävän osion ohjelman yläreunassa olevasta pudotuslistasta. Esimerkkinä tutkitaan edellä mainittua tuotedokumenttien kenttien rajausta. Taulukko on edelleen Delphin vakiotaulukosta tehty paranneltu versio, muut komponentit ovat Delphissä vakiona olevia.



Kuvio 27. Järjestelmän ohjaustietojen ylläpito



Kuvio 28. Ylläpitonäkymän valinta

Kun käyttäjä on valinnut listalta kohdan Document-type Fields (kuvio 28), taulukkoon avautuu kuvion 27 mukainen näkymä. Taulukko täytetään SQL Serverillä olevalla Stored Procedurella, ViewDoctypeVisibleFields-aliohjelmalla, joka parametrinään saa näytettävien taulujen nimet. Tämä tallennettu sql-aliohjelma kokoaa taulukkoon DocTypeFields-taulukon sisällön kuvion 27 mukaisella tavalla.

Esimerkki 6. Stored procedure ylläpitonäkymän koostamiseen

```
procedure dbo.ViewDoctypeVisibleFields(@TableFilter varchar(150)) as
begin
  DECLARE @SQL varchar(2000)
  DECLARE @SubSQL varchar(150)
  DECLARE @loop int
  DECLARE @loopCount int

  SELECT TypeID, TypeDescription INTO #DocTypeTemp FROM FieldListContents where ListID =
1
  CREATE TABLE #temp (DocTypeId smallint, [FieldName] varchar(25) collate
SQL_Latin1_General_CP1_CI_AS)
  INSERT INTO #temp SELECT DocTypeId, [FieldName]  FROM DocTypeFields

  CREATE TABLE #UserTableNames (TableName varchar(200) collate
SQL_Latin1_General_CP1_CI_AS, TableOrder int)
  INSERT INTO #UserTableNames VALUES  ('GeneralInformation',1)
  INSERT INTO #UserTableNames VALUES  ('ComponentInformation',2)
  INSERT INTO #UserTableNames VALUES  ('TechnicalData',3)
  INSERT INTO #UserTableNames VALUES  ('PackingAndMarking',4)
  INSERT INTO #UserTableNames VALUES  ('ReferencedTechnicalDocuments',5)
  INSERT INTO #UserTableNames VALUES  ('QualityInspection',6)
  INSERT INTO #UserTableNames VALUES  ('Drawings',7)

  SET @loop = (SELECT min(TypeID) FROM #DocTypeTemp)
  SET @loopCount = (SELECT max(TypeID) FROM #DocTypeTemp)
  SET @SQL =  'select so.name as Category, sc.name as Fieldname '

  while @loop <= @loopCount
  begin
    SET @SubSQL =   ' ,(SELECT ''X'' from #temp t WHERE t.FieldName = sc.Name AND
        t.DocTypeID = '+convert(varchar,@loop)
        +') as ['+ (SELECT TypeDescription FROM #DocTypeTemp WHERE TypeID = @loop)+']'
```

```
    SET @SQL =  @SQL + @SubSQL
    SET @loop = @loop + 1
  end

  if @tableFilter = 'all'
  begin
   SET @SQL =  @SQL  +' from syscolumns sc, sysobjects so, #UserTableNames tn where sc.id =
   so.id and so.[name] in (SELECT TableName FROM #UserTableNames) AND so.[name] =
    tn.TableName
    AND (sc.Name not in (''ProductCode'',''DocumentType'',''Version'')) ORDER BY tn.TableOrder'
  end
  else
  begin
   SET @SQL =  @SQL  +' from syscolumns sc, sysobjects so, #UserTableNames tn where sc.id =
   so.id and so.[name] in ('+@tableFilter+')  AND so.[name] = tn.TableName
   AND (sc.Name not in (''ProductCode'',''DocumentType'',''Version'')) ORDER BY tn.TableOrder'
  end
  exec (@SQL)
end
```

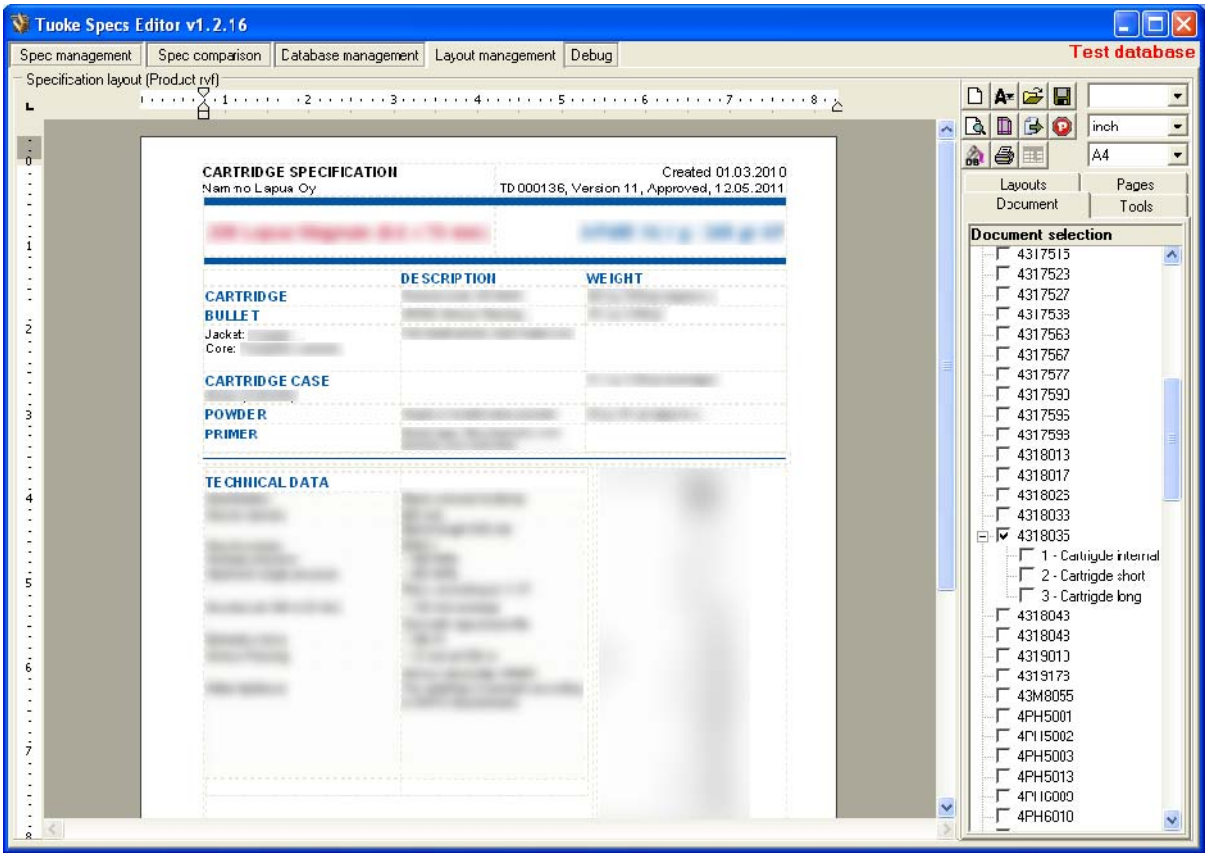
Ohjelmakoodissa taulukon solujen CellClick-tapahtumaan on kytketty alla oleva koodi.

Esimerkki 7. CellClick-tapahtuman koodi ylläpitonäkymässä

```
Procedure dbgrid2.CellClick(Column: TColumn);
var
   updQuery : TADOQuery;
begin
  if (cmbManageTaulu.Text = 'Document-type fields') then
  begin
     UPDQuery := TADOQuery.Create(nil);
     UPDQuery.Connection := connSpecmanage;


     If Uppercase(Column.Field.AsString) = 'X' then
     begin
        UPDQuery.SQL.Text := 'DELETE FROM DocTypeFields WHERE '
                 +' FieldName = '''
                 + dsKaliiperit.DataSet.FieldByName('Fieldname').AsString +
                 ''' AND DocTypeID = '''+
                 inttostr(Column.Field.Index-1)+'';
        UPDQuery.ExecSQL;
      end
      else if Column.Field.IsNull then
      begin
        UPDQuery.SQL.Text := 'INSERT INTO DocTypeFields (DocTypeID, TableName,
                             FieldName) VALUES ('+ inttostr(Column.Field.Index-1) +', '''+
                             dsKaliiperit.DataSet.FieldByName('Category').AsString
                             +''' ,'''+
                             dsKaliiperit.DataSet.FieldByName('Fieldname').AsString+''')';
        UPDQuery.ExecSQL;
      end;

      dbgrid2.RefreshData;

   end;
end;
```

Taulukon solua painettaessa tutkitaan ensin ollaanko tuotekenttien rajausnäkymässä eli onko pudotuslistassa valittuna ’Document-type fields’. Sen jälkeen luodaan dynaamisesti ADOQuery-luokka. ADOQuery on luokka, jolla voidaan tehdä sql-kyselyitä ja muita sql-operaatioita. Jos painetun kentän sisältönä oli ’X’, niin sitten luodaan ADOQueryyn SQL Delete -lause, jolla poistetaan ohjaustaulusta kyseistä kenttää tarkoittava merkintä. Jos kenttä taas oli painettaessa NULL, Column.Field.IsNull, eli taulussa ei ole merkintää tälle tietokantakentälle, luodaan SQL Insert -lause. Tällä Insert-lauseella ohjaustauluun syötetään merkintä taulun solua vastaavan dokumenttityypin, tuotetaulun ja kentännimen mukaisilla tiedoilla.

4.3.8 Tuotedokumenttien ylläpito



Kuvio 29. Tuotedokumenttien ylläpito

Ohjelman neljäs moduuli on tuotetietojen ylläpito. Se on ohjelman ehkä isoin yksittäinen kokonaisuus ja se koostuu kahdesta eri pääosasta ja niiden ympärille rakennetuista toiminnoista ja ominaisuuksista.

Moduulin osat ovat kuviossa 29 keskellä oleva RichView-editori komponentti ja siitä niin sanottu Scaled-versio, joka mahdollistaa dokumentin skaalauksen, esikatselun ja muita tekstinkäsittelystä tuttuja toimintoja. Toinen ostettu komponentti on LLPdfLib, jolla ei ole näkyvää komponettia, mutta sen avulla ikkunassa avattuna oleva dokumentti muutetaan pdf-muotoon ja tallennetaan joko tietokantaan pdf- tai rvf-tiedostoksi jonnekin käyttäjän määrittelemään hakemistoon.

Käyttäjällä olevat työkalut jakautuvat kahteen osaan: Ohjelmaikkunan oikeasssa yläreunassa olevaan toimintokenttään, jossa on dokumenttiin liittyviä perustoimintoja, kuten muun muassa avaa, tallenna, tulosta, julkaise ja täytä tiedot tietokannasta (kuvio 30). Niiden lisäksi on kolme pudotuslistaa, joiden avulla voidaan muuttaa dokumentin suurennosta, paperikokoa ja dokumentin mitoissa käytettyä mittayksikköä. Lisäksi löytyy nappi, jonka avulla voidaan asettaa niin sanottu avausmoodi eli mitä dokumentteja järjestelmä oletusarvoisesti lähtee hakemaan tietokannasta käyttäjän painaessa nappia avaa. Vaihtoehdot ovat A – automaattinen, L – Oletuspohja, R – Rvf-tiedosto eli tallennettu tuotedokumentti, ja F – Paikallinen tiedosto. Automaattinen valinta tutkii ensin löytyykö tallennettua tuotedokumenttia. Jos sitä ei löydy, avataan tuotedokumentin tyyppiin liittyvä oletuspohja. Jos sitäkään ei löydy, näytetään käyttäjälle dialogi, jonka avulla voi avata paikallisen tiedoston.

Kuvio 30. Dokumenttien ylläpidon perustoiminnot

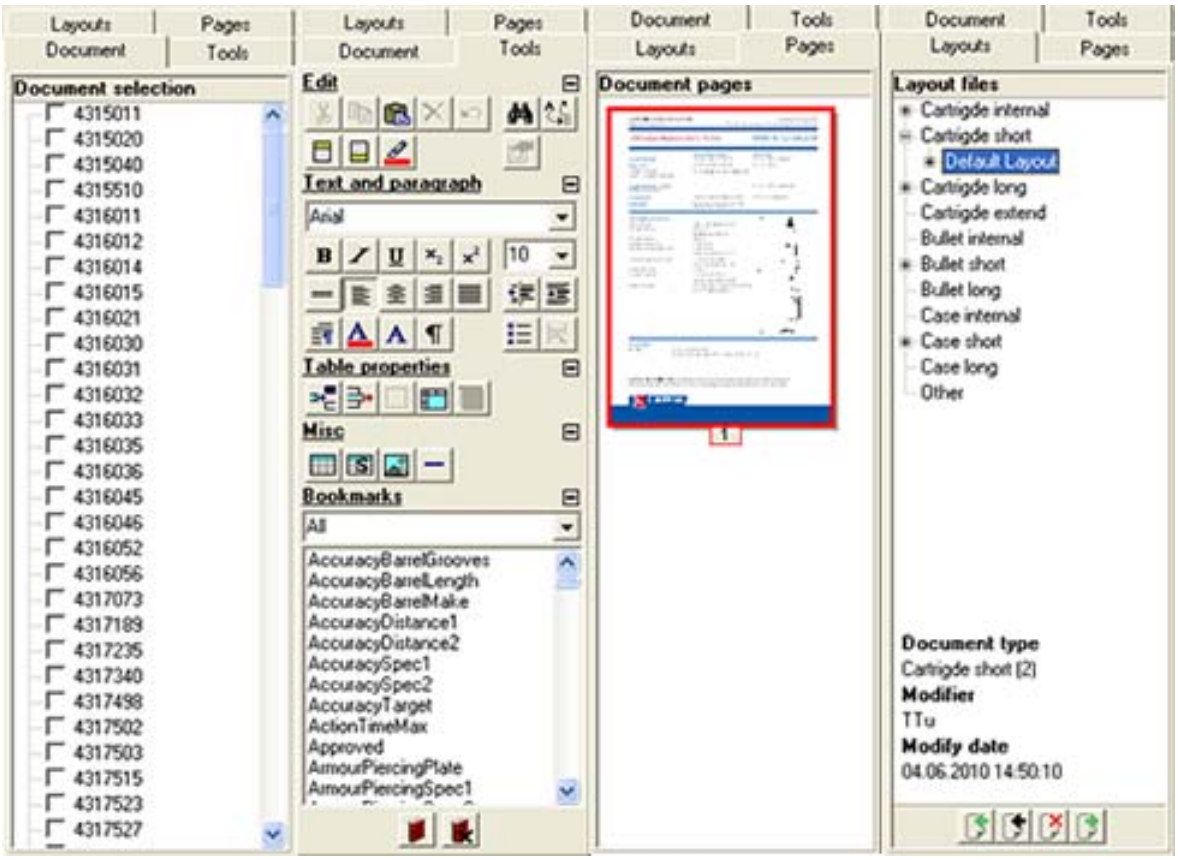
Tuotedokumentin avaus tietokannasta on toteutettu ohjelmakoodissa seuraavasti: Aluksi on luodaan dynaamisesti queryFiles-objekti, joka on tyypiltään TADOQuery.

Sen jälkeen luodaan sille Sql-lause avattavan dokumentin tuotekoodin, dokumenttityypin ja dokumentin version avulla. Jos avaintiedoilla on tallennettu tuotedokumentti, recordCount = 1, niin kentän ProductRvf sisältö tallennetaan muistivirtaan streamRvfFile. Tallennuksen jälkeen muistivirran osoitin, position, siirretään alkuun ja editoriin, ActiveSRVE, ladataan muistivirtaan tallennettu dokumentti.

Esimerkki 8. Tuotedokumentin avaus tietokannasta

```
queryFiles := TADOQuery.Create(nil);
queryFiles.Connection := connSpecmanage;
queryFiles.SQL.Text := 'SELECT * FROM ProductRvfs WHERE ProductCode ='''+
DocProductCode+ ''' AND DocumentType = '+DocDocumentType+' AND Version = '+
                   '(SELECT Max(Version) FROM ProductRvfs WHERE ProductCode='''+
                   DocProductCode+''' AND DocumentType='+DocDocumentType+')';
queryFiles.Open;
if queryFiles.RecordCount = 1 then
begin
   TBlobField(queryFiles.FieldByName('ProductRvf')).SaveToStream(streamRvfFile);
   streamRvfFile.Position := 0;
   ActiveSRVE.LoadRVFFromStream(streamRvfFile);
   ActiveSRVE.Format;
end;
```

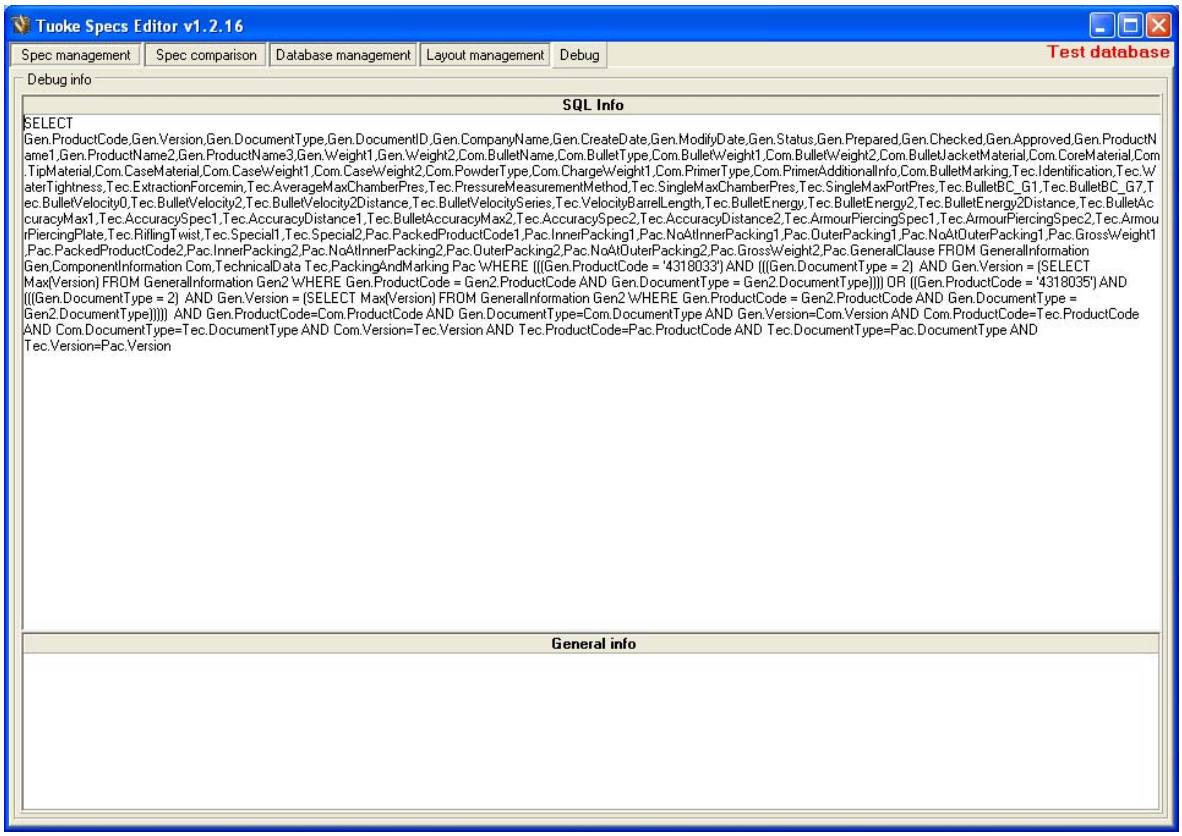
Kuviossa 30 esitetyn toimintokentän alla on ohjelmassa kokoelma, jossa on välilehdille jaoteltu lisää dokumentin editoinnissa tarvittavia työkaluja. Ensimmäisen välilehden kautta voidaan valita avattava dokumentti, dokumenttityyppi ja versio. Toisella välilehdellä on tekstinkäsittelyistä tutut työkalut, joiden alta löytyy vielä niin sanotut kirjanmerkit. Kirjanmerkkien avulla dokumentille voidaan luoda kenttiä, jotka täytetään tietokannan kenttien sisällön mukaan. Kolmas välilehti sisältää dokumentin sivut sivunäkymänä ja tämän avulla käyttäjä voi hypätä eri sivuille dokumentissa. Neljäs välilehti on tarkoitettu oletuspohjien hallintaan ja sen avulla voidaan oletuspohjia ladata tietokantaan, poistaa niitä, tallentaa niitä tiedostoksi sekä hakea niitä editoitavaksi ohjelmaan. Esimerkki tyhjästä dokumenttipohjasta löytyy liitteenä 3.



Kuvio 31. Dokumenttien ylläpidon lisätoiminnot

4.3.9 Debug- ja toimintonäkymät

Ohjelman viides ja viimeinen osa on moduuli, jossa näkyvät mahdolliset debug- ja toimintonäkymät. Tämän kautta käyttäjä voi virhetilanteessa tarkistaa, jos ohjelma antaa lisätietoa virheen laadusta. Debug-näkymän kautta voi myös tarkistaa edellisen sql-lauseen sisällön, josta voi virhetilanteessa olla myös hyötyä. Tällä moduulilla ei ole omia toimintoja, mutta käytettäessä muita moduuleita, tämän moduulin sisältö päivittyy ohjelman toimintojen mukaan.

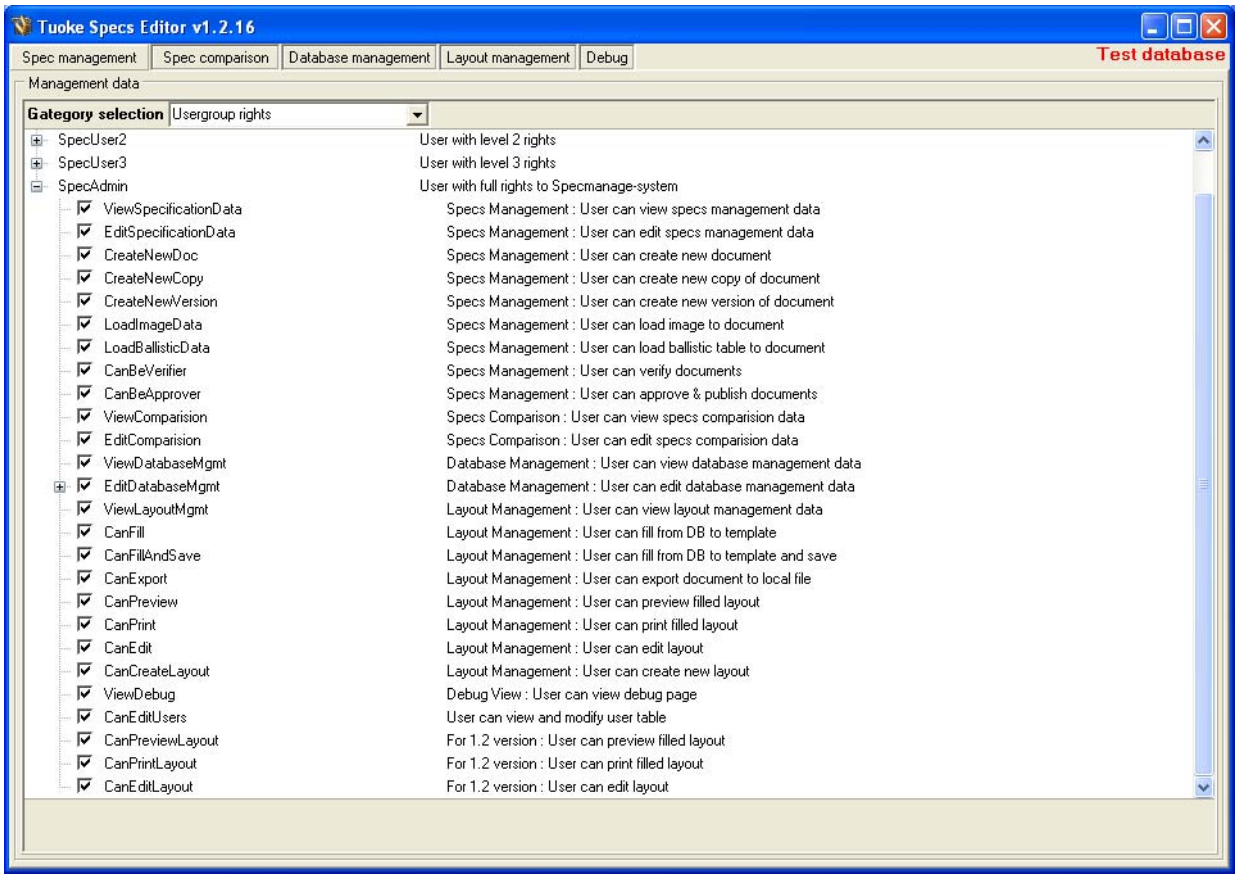


Kuvio 32. Debug- ja toimintonäkymä

4.3.10 Käyttäjäoikeuksien hallinta

Järjestelmän ohjaustietojen hallinnan kautta voidaan muokata myös ohjelman käyttöoikeuksia eri käyttäjäryhmille. Ohjaustietojenhallinta moduulista lisää löytyy kappaleesta 4.3.3.

Käyttäjäoikeudet rakentuvat Windows-ympäristöstä tutulla tavalla. Järjestelmä sisältää käyttäjäryhmiä, joiden avulla käyttäjäoikeuksia hallinnoidaan. Käyttäjät kuuluvat eri käyttäjäryhmiin ja saavat käyttöönsä oman ryhmänsä mukaiset oikeudet käyttäessään järjestelmää. Erilaisia käyttäjäoikeuksia on noin 20 kappaletta ja niiden avulla voidaan hallita esimerkiksi sitä, voiko käyttäjä muokata vai vain nähdä tuotedokumentin tiedot. Kuviossa 33 on esimerkkinä järjestelmän pääkäyttäjällä olevat oikeudet.



Kuvio 33. Käyttäjäoikeuksien hallinta

Käyttäjäoikeusnäkymän puunäkymään on OnClick-tapahtumaan liitetty seuraavanlainen ohjelmakoodi.

Esimerkki 9. Käyttäjäoikeusnäkymän puunäkymän OnClick-ohjelmakoodi

```
procedure Tlomake1.vrTreeClick(Sender: TObject);
var
   P:TPoint;
   i : integer;
   queryRightsChange : TADOQuery;
   NodeData, NodeDataParent : ^vrTreeData;
   Node, NodeParent : PVirtualNode;
   UGroupID : string;
   ht : THitInfo;
   hp : THitPositions;
begin
   GetCursorPos(P);
   P := vrTree.ScreenToClient(P);
   vrTree.GetHitTestInfoAt(P.X,P.Y,true,ht);
   hp := ht.HitPositions;
   if (hiOnStateIcon in hp) and (vrTree.Tag = 0) then
   begin
     NodeData := vrTree.GetNodeData(ht.HitNode);
     if NodeData.StateIndex = 1 then
       NodeData.StateIndex := 2
```

```
else if NodeData.StateIndex = 2 then
 NodeData.StateIndex := 1;

 vrTree.Repaint;
 queryRightsChange := TADOQuery.Create(nil);
 queryRightsChange.Connection := connSpecmanage;
 queryRightsChange.SQL.Text := '';

  if (NodeData.NodeLevel = 1) and (NodeData.StateIndex = 2)  then
 begin
   NodeParent := ht.HitNode.Parent;
   NodeDataParent := vrTree.GetNodeData(NodeParent);

   queryRightsChange.SQL.Add('if not exists('+
                    'SELECT * FROM Specmanage.dbo.UserGroupRights '+
    'WHERE UserGroupID = '+NodeDataParent.NodeValue+
    ' AND UserRightID = '+NodeData.NodeValue+')');
   queryRightsChange.SQL.Add('begin');
   queryRightsChange.SQL.Add('INSERT INTO Specmanage.dbo.UserGroupRights '+
    ' (UserGroupID, UserRightID) VALUES ('+NodeDataParent.NodeValue+
    ', '+NodeData.NodeValue+')');
   queryRightsChange.SQL.Add('end');

   queryRightsChange.ExecSQL;

   saveEvent('Add userpermission','UserGroupID='+
                        NodeDataParent.NodeValue+
                        ';UserRightID='+NodeData.NodeValue,'');
 end
 else if (NodeData.NodeLevel = 1) and (NodeData.StateIndex = 1)  then
 begin
   NodeParent := ht.HitNode.Parent;
   NodeDataParent := vrTree.GetNodeData(NodeParent);

   if ((NodeData.TextCol1 = 'CanEditUsers') and
   (NodeDataParent.TextCol1 = 'SpecAdmin')) then
   begin
    NodeData.StateIndex := 2;
    vrTree.Refresh;
 end
 else if not ((NodeData.TextCol1 = 'CanEditUsers') and
 (NodeDataParent.TextCol1 = 'SpecAdmin')) then
 begin

   queryRightsChange.SQL.Add('DELETE FROM Specmanage.dbo.UserGroupRights '+
    ' WHERE UserGroupID = '+NodeDataParent.NodeValue+
    ' AND UserRightID = '+NodeData.NodeValue);

   queryRightsChange.ExecSQL;

   saveEvent('Delete userpermission','UserGroupID='+
                        NodeDataParent.NodeValue+
                        ';UserRightID='+NodeData.NodeValue,'');
  end;
end
else if (NodeData.NodeLevel = 2) and (NodeData.StateIndex = 2)  then
begin
  NodeParent := ht.HitNode.Parent.Parent;
  NodeDataParent := vrTree.GetNodeData(NodeParent);
```

```
      queryRightsChange.SQL.Add('if not exists('+
                        'SELECT * FROM Specmanage.dbo.UserGroupDBManagementRights '+
       'WHERE UserGroupID = '+ NodeDataParent.NodeValue+
       ' AND TableId = '+NodeData.NodeValue+')');
      queryRightsChange.SQL.Add('begin');
      queryRightsChange.SQL.Add('INSERT INTO Spec-
manage.dbo.UserGroupDBManagementRights '+
       ' (UserGroupID, TableID) VALUES ('+NodeDataParent.NodeValue+
       ', '+NodeData.NodeValue+')');
      queryRightsChange.SQL.Add('end');
      queryRightsChange.ExecSQL;

      saveEvent('Add userpermission (table)','UserGroupID='+
                           NodeDataParent.NodeValue+
                           ';TableID='+NodeData.NodeValue,'');
    end
    else if (NodeData.NodeLevel = 2) and (NodeData.StateIndex = 1)  then
    begin
      NodeParent := ht.HitNode.Parent.Parent;
      NodeDataParent := vrTree.GetNodeData(NodeParent);

      queryRightsChange.SQL.Add('DELETE FROM Spec-
manage.dbo.UserGroupDBManagementRights '+
       ' WHERE UserGroupID = '+NodeDataParent.NodeValue+
       ' AND TableId = '+NodeData.NodeValue);

      queryRightsChange.ExecSQL;
      saveEvent('Delete userpermission (table)','UserGroupID='+
                           NodeDataParent.NodeValue+
                           ';TableID='+NodeData.NodeValue,'');
    end;

    queryRightsChange.Close;
    queryRightsChange.Free;
  end;
end;
```

Aluksi luodaan dynaamisesti TADOQuery, queryRightsChange, jonka avulla edellä mainittuihin tauluihin tehdään muutoksia käyttäjän tekemien valintojen mukaan. Sitten tutkitaan mitä puunäkymän valintaa on painettu eli onko kyseessä käyttöoikeus vai taulunmuokkaus-oikeus. Tarkistetaan myös halutaanko lisätä käyttöoikeus vai poistaa jo olemassa oleva eli tutkitaan onko puunäkymän valintaruutu valittuna vai ei käyttäjän painamalla kohdalla. Käyttäjäryhmien käyttöoikeudet ovat taulussa UserGroupRights ja vastaavat käyttäjäryhmien taulujenmuokkausoikeudet ovat taulussa UserGroupDBManagementRights. Edellisten tarkistusten jälkeen luodaan tarvittava Sql-lause queryRightsChange-objektille ja päivitetään sen avulla käyttäjäoikeudet järjestelmän ohjaustauluihin.

5 VALMIIN JÄRJESTELMÄN ARVIOINTI

Järjestelmä saavutti suurimmaksi osaksi sille asetetut tavoitteet. Joitakin alun perin suunniteltuja ominaisuuksia jäi toteuttamatta, mutta ne eivät ole järjestelmän toiminnan kannalta sellaisia, ettei järjestelmä voisi toimia ilmankin niitä. Lisäksi järjestelmää voidaan hyvin laajentaa jatkossa uusilla tai toteutumatta jääneillä ominaisuuksilla, jos tarvetta joskus on.

Verrattuna aikaisempaan tuotedokumenttien hallinnointi on uuden järjestelmän avulla selvästi tehokkaampaa ja helpompaa. Dokumentteihin sisältyvät tiedot ovat erillään tuotedokumenttien visuaalisesta rakenteesta, joten tietoja ja visuaalista rakennetta voidaan molempia muuttaa erikseen. Tietojen vertailu on nyt paljon helpompaa, koska tiedot voidaan hakea suoraan tietokannasta taulukkoon ja taulukoon tuotavia kenttiä voidaan vielä rajata tarpeen mukaan. Myös visuaalista ilmettä uusittiin, tuotedokumentit saivat aivan uuden ilmeen. Jatkossa visuaalisten muutosten tekeminen käy lisäksi myös helpommin, koska eri tuotedokumenttityypeille on määritelty oletuspohjat, joiden muuttaminen vaikuttaa heti kaikkiin kyseistä pohjaa käyttäviin dokumentteihin.

Alkuvaiheessa tuotedokumenttien hallintajärjestelmän rakentaminen vaikutti hitaammalta ja suuritöisemmältä projektilta, mitä osasi odottaa. Sitä se toisaalta tietysti myös oli. Jos tulevan järjestelmän mahdollistamat edut eivät hahmottuneet tarpeeksi selkeästi, alkuvaiheen suunnittelun ja määrittelyn aikana voi jopa tuntua, että onko dokumenttien hallintajärjestelmän rakentamiseen käytetty aika ja vaiva sen arvoista. Mutta tehty työ kannattaa, sillä se mikä alun ajassa ja vaivassa menetetään, saadaan takaisin toimivan järjestelmän tehokkuutena ja helppoutena. Saavutettuja etuja ei välttämättä voida mitata edes kovin helposti. Voitaneen sanoa, että entisen järjestelmän aiheuttamat pienetkin virheet olisivat voineet aiheuttaa pahimmillaan työmäärän, jonka kustannuksilla olisi mahdollisesti rakennettu jopa huomattava osa toteutettua uutta dokumenttien hallintajärjestelmää. Nyt uuden järjestelmän avulla kyseisten virheiden mahdollisuus on minimoitu siirtämällä kaikki tiedot kokonaisuudessaan tietokantaan, jolloin niiden tarkistus ja kontrollointi jo syöttövaiheessa tai myöhemmin on paljon helpompaa.

LÄHTEET

Anttila, J. 2001. Dokumenttien hallinta. Helsinki: Edita Oyj.

Heikkinen, A. .1994. Dokumenttien hallinta tuottavuuden välineenä. Espoo: Dipoli

Haikala, I. & Märijärvi, J. .2004. Ohjelmistotuotanto. 10. painos. Helsinki: Talentum Media Oy.

Hovi, A. .2004. SQL-opas. 1. painos. Jyväskylä: Docendo Finland Oy

Swan, T. 1999. Delphi 4. 1. painos. Jyväskylä: Teknolit Oy

Embarcadero. 2011. Delphi XE2 | RAD Application Development Software. [www-dokumentti]. Embarcadero Technologies Inc. [Viitattu 21.11.2011]. Saatavissa: http://www.embarcadero.com/products/delphi

TRichview. 2011. RichView Components for Delphi and C++Builder. [www-dokumentti]. TRichview. [Viitattu 12.10.2011]. Saatavissa: http://www.trichview.com/

Llionsoft. 2011. Welcome to llionsoft. [www-dokumentti]. Llionsoft. [Viitattu 12.10.2011]. Saatavissa: http://www.llion.net/

Microsoft. 2011. Active Directory Server. [www-dokumentti]. Microsoft. [Viitattu 18.10.2011]. Saatavissa: http://www.microsoft.com/en-us/server-cloud/windows-server/active-directory-overview.aspx

LIITTEET

Liite 1: Suunnittelumuistio järjestelmän kehitysvaiheesta

Liite 2: Kaavio dokumentin hyväksymismenettelystä

Liite 3: Tuotedokumentin tyhjä pohja

Liite 4: Tuotetietotaulujen rakenne

Liite 5: Järjestelmän ohjaustaulujen rakenne

Liite 1. Suunnittelumuistio järjestelmän kehitysvaiheesta

TuokeSPECS ohjelma ja spesifikaatiotietokanta

Paikka: Hylsyvintti
Aika: 3.6.2010
Osanottajat: MTU, TTU, MPA

1 Tilannekatsaus

Tietokanta määritellyiltä osin valmis ja käytössä tuotantopuolella ja testipuolella Tietokannan kuvasta tehty, mutta tarve listata kaikki taulut kuvaukseen.

Ohjelma käytössä ja spesifikaation pystyy luomaan:

- Lyhyt patruunaspesifikaatio (lopullinen versio)
- Lyhyt luotispesifikaatio (muutoksia voi tulla)
- Lyhyt hylsyspesifikaatio (muutoksia voi tulla)

2 Kehitystarpeet

Tulevan kesän aikana pyritään tekemään seuraavia muutoksia listatussa järjestyksessä:

2.1 Hyväksytyn version lukitseminen

Hyväksyttyä dokumenttia ei voi muuttaa hyväksynnän tallennuksen jälkeen. (tehty 28.6)

2.2 Käyttäjien ylläpito ja oikeudet

Määritellään tietokantaan taulu, jossa kaikkien käyttäjien tiedot:

- Nimi
- Lyhenne, esim. TTu
- Sähköposti
- Oikeudet eli käyttäjäryhmä

Oikeudet taulu erikseen (jos säädettävä)

Näiden avulla pyritään hallitsemaan näkymiä ja muutosoikeuksia. Liittyy kohtaan 2.5

2.3 Dokumenttien hyväksyntärutiini

Karkealla tasolla on määritelty rutiini, minkä tarkoituksena on dokumentin viimeistelyn jälkeen saattaa se tarkastettavaksi ja hyväksyttäväksi, sekä ilmoittaa siitä ko. toimenpiteen suorittajalle. Em. henkilölle on myös tarpeen lähettää tai osoittaa tarvittavat tiedot toimenpitene suorittamiseksi.

Rutiini koko dokumentin luontiketjun aikana on dokumentin: *"SPECIFICATION Upgrade v8.doc/ Kaavio 3: Document creation process & Approval routine"* – mukainen.

Katkoviivoituksella erotettu hyväksuntärutiinin muodostama osa.

2.4 Käyttöliittymän kehitys

TuokeSPECS-editorin käyttöliittymään on suunniteltu seuraavien alakohtien mukaiset kehityskohteet:

2.4.1 Template check

Tarkoitus kuvattu dokumentissa: *"SPECIFICATION Upgrade v8.doc/Kaavio 1: Forming of published information"*
Toiminto testaa onko pohja muuttunut edellisestä päivityskerrasta ja ehdottaa uuden pohjan ottamista käyttöön.

2.4.2 PDF creation process

Automaattinen PDF-tiedoston luonti (ei CutePDF writer) ja tallennus ennalta määrättyyn paikkaan auotmaattisesti generoidulla tiedostonimellä.

2.4.3 Ilmoitus toiminnon käynnissä olemisesta

Mallia "tiimalasi" tai ilmoitus kun prosessi on valmis.

2.4.4 Pohjan muokkaus

Työkalut layout-puolelle pohjan rvf-template:n tekemiseksi. Tavoitteena "word"-malliset työkalut.

2.5 Kenttien näkymän rajaus

Rajaus eri käyttäjäprofiileille, esim. 5 tasoa
Oikeudet seuraaviin (voidaan päivittää listaa myöhemmin)

1. View **Specs management** page
2. Modify document data
3. Create new document

4. Create ne copy
5. Create new version
6. Load image
7. Load ballistic table
8. Can be creator of document (can send it for verify)
9. Can be verifier
10. Can be approver
11. View **Specs comparison** page
12. View Specs comparison page with open fields (edit)
13. View **Database management** page
14. Modify database management tables (separate)
15. View **Layout management** page
16. Can fill DB to template
17. Can fill DB to template and save
18. Can Preview filled layout
19. Can Print filled layout
20. Can edit layout
21. Can create new layout
22. View Debug
23. Can view / modify user table

Oikeudet seuraavasti (voidaan päivittää taulukkoa myöhemmin)

Taso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Viewer	x														x								
User 1	x		x		x				x		x		x		x	x		x					
User 2	x		x	x	x	x	x	x	x		x		x		x	x	x	x					
User 3	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
Admin	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

2.6 Tietokannan solujen automaattinen kopiointi

Tietokannassa määriteltyjen solujen automaattinen kopiointi dokumenttityypiltä toiselle.

Lähdetyyppi	Kopiointi
1	2, 3, 4
5	6, 7
8	9, 10

Rutiinin kuvaus ja rajoitukset esitetty: *"SPECIFICATION Upgrade v8.doc/ Kaavio 2: Database update routine"*

2.7 Uudet layout-pohjat tulevaisuudessa:

Pohjien rakentaminen spesifikaatiotyypeille:

- Tyyppi 3, Pitkä patruunaspesifikaatio
- Tyyppi 4, Erityispitkä patruunaspesifikaatio

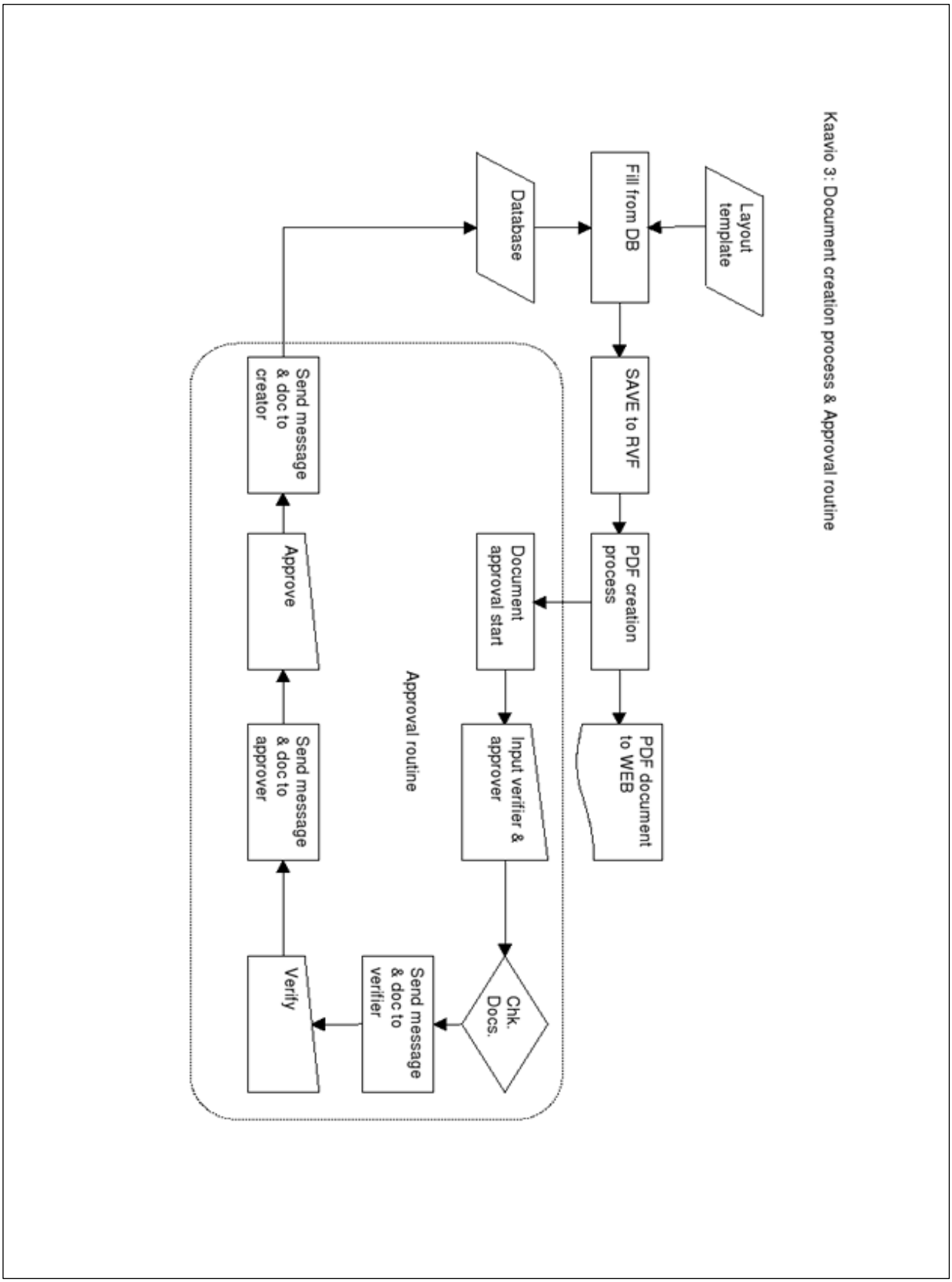
- Tyyppi 7, Luodin pitkä spesifikaatio
- Tyyppi 10, Hylsyn pitkä spesifikaatio

Jatkossa mietitään periaate sisäisten ladunvalvontaohjeiden käsittelystä (ohjeet PDM:ssä) ja pohjat rakennetaan sen jälkeen.

- Tyyppi 1, Sisäinen spesifikaatio
- Tyyppi 5, Luodin laadunvalvontaohje
- Tyyppi 8, Luodin laadunvalvontaohje

Näiden tarpeen ajankohta määritellään erikseen. Valmius em. spesifikaatioiden datan käsittelylle on tietokannassa ja käyttöliittymässä.

Liite 2. Kaavio dokumentin hyväksymismenettelystä



Liite 3. Tuotedokumentin tyhjä pohja

CARTRIDGE SPECIFICATION #Bookmark1#
#CompanyName# #DocumentID#, #Bookmark2#

#ProductName1# #ProductName2# **#ProductName3#**

	DESCRIPTION	WEIGHT
CARTRIDGE		
BULLET		
#BulletJacketMaterial# #CoreMaterial# #TipMaterial#		
CARTRIDGE CASE		
#CaseMaterial#		
POWDER		
PRIMER		

TECHNICAL DATA

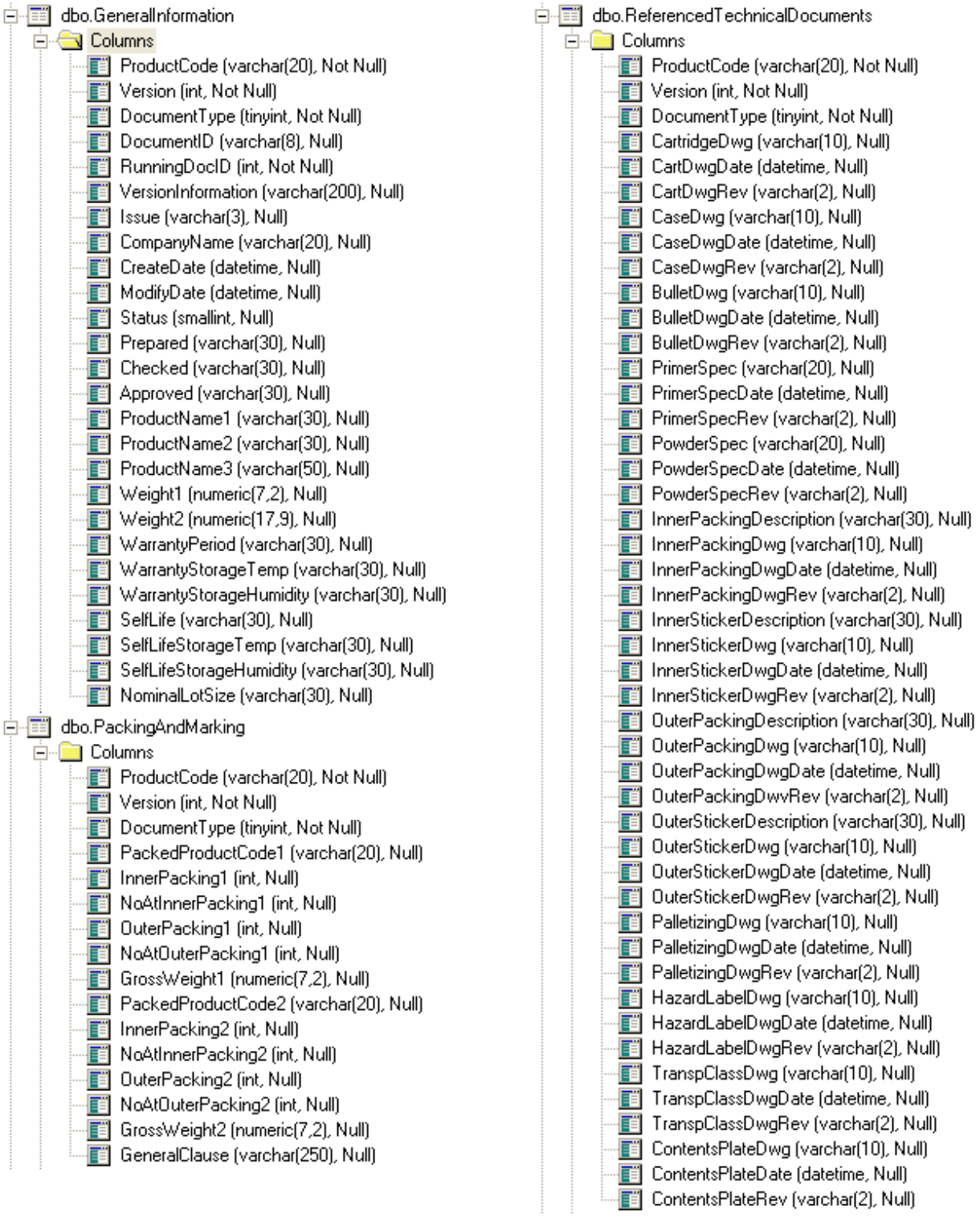
CompDrawing

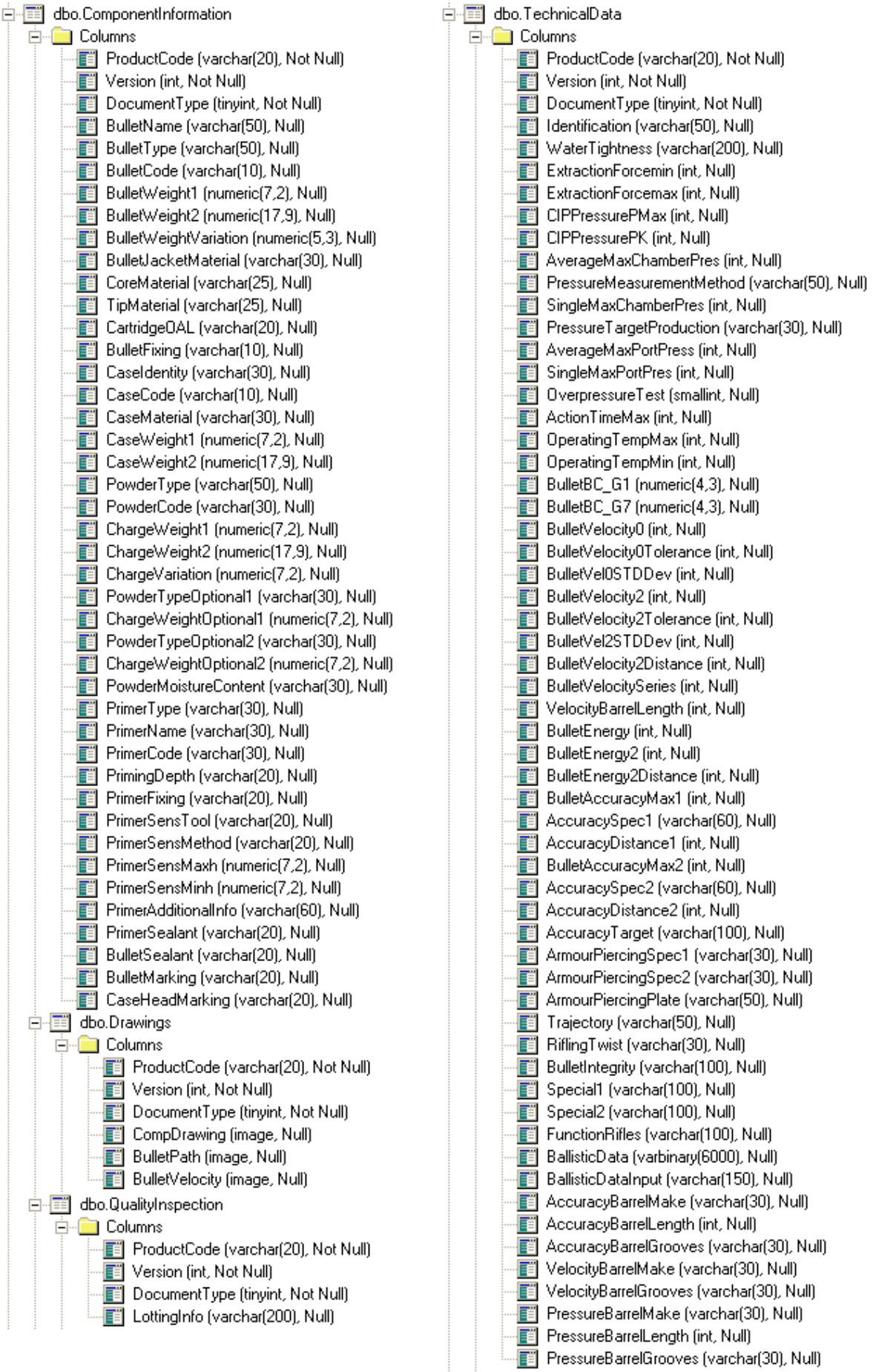
PACKING

LAPUA AMMUNITION is packed according to the newest regulations for sea transport (IMDG-code) and according to the IATA dangerous goods regulations for air carriage.

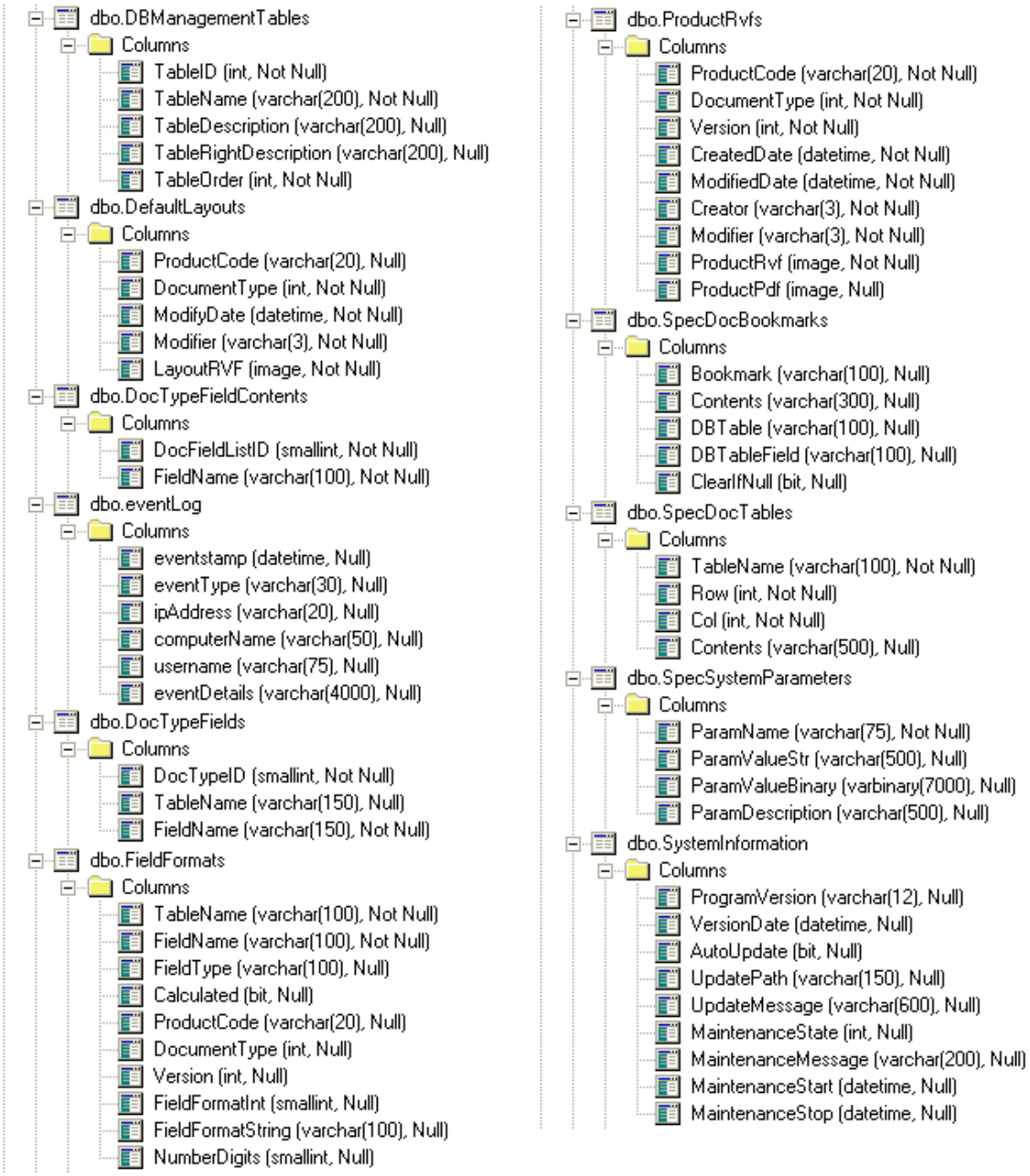
LAPUA

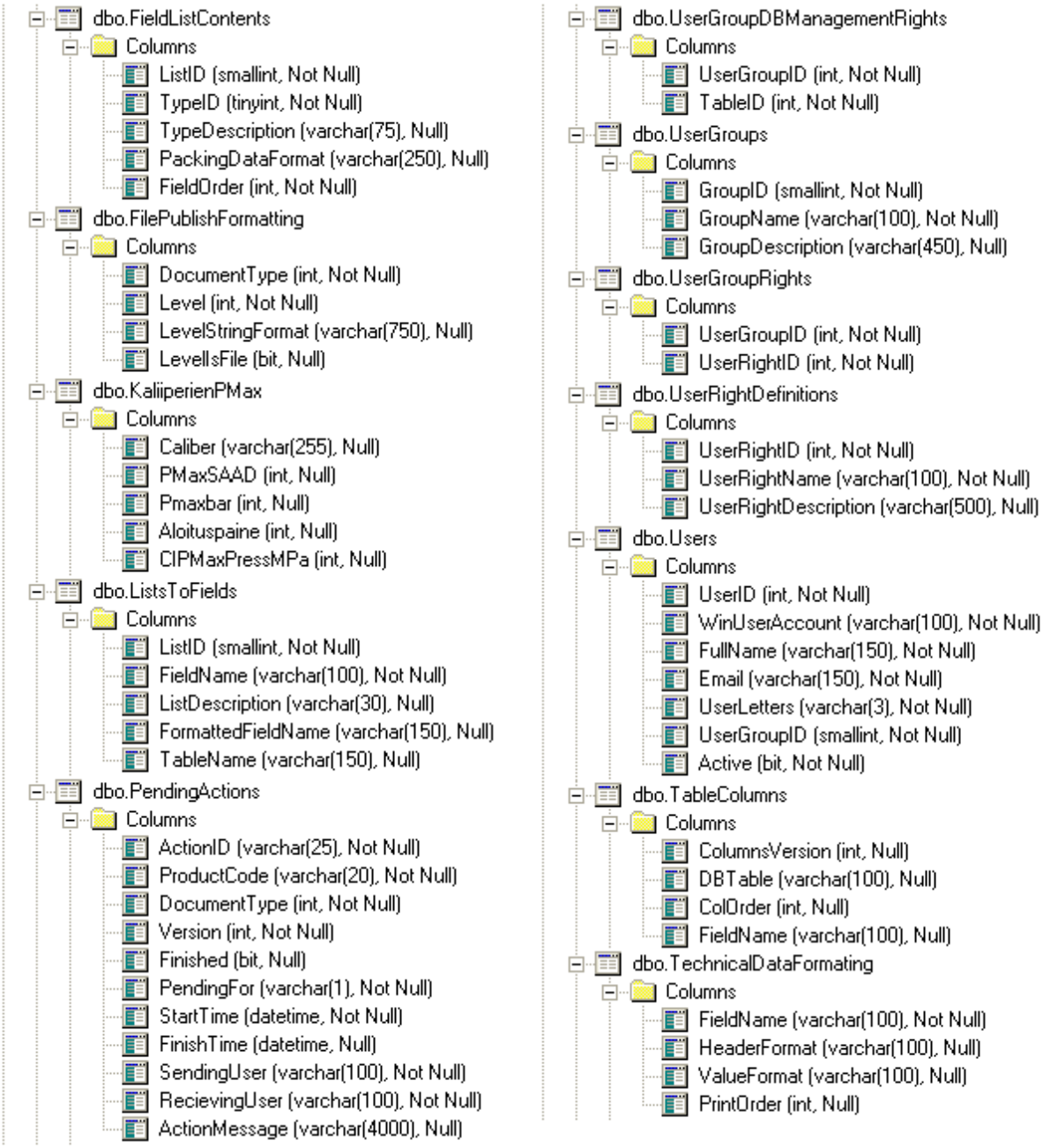
Liite 4. Tuotetietotaulujen rakenne




dbo.ComponentInformation
Columns
ProductCode (varchar(20), Not Null)
Version (int, Not Null)
DocumentType (tinyint, Not Null)
BulletName (varchar(50), Null)
BulletType (varchar(50), Null)
BulletCode (varchar(10), Null)
BulletWeight1 (numeric(7,2), Null)
BulletWeight2 (numeric(17,9), Null)
BulletWeightVariation (numeric(5,3), Null)
BulletJacketMaterial (varchar(30), Null)
CoreMaterial (varchar(25), Null)
TipMaterial (varchar(25), Null)
CartridgeOAL (varchar(20), Null)
BulletFixing (varchar(10), Null)
CaseIdentity (varchar(30), Null)
CaseCode (varchar(10), Null)
CaseMaterial (varchar(30), Null)
CaseWeight1 (numeric(7,2), Null)
CaseWeight2 (numeric(17,9), Null)
PowderType (varchar(50), Null)
PowderCode (varchar(30), Null)
ChargeWeight1 (numeric(7,2), Null)
ChargeWeight2 (numeric(17,9), Null)
ChargeVariation (numeric(7,2), Null)
PowderTypeOptional1 (varchar(30), Null)
ChargeWeightOptional1 (numeric(7,2), Null)
PowderTypeOptional2 (varchar(30), Null)
ChargeWeightOptional2 (numeric(7,2), Null)
PowderMoistureContent (varchar(30), Null)
PrimerType (varchar(30), Null)
PrimerName (varchar(30), Null)
PrimerCode (varchar(30), Null)
PrimingDepth (varchar(20), Null)
PrimerFixing (varchar(20), Null)
PrimerSensTool (varchar(20), Null)
PrimerSensMethod (varchar(20), Null)
PrimerSensMaxh (numeric(7,2), Null)
PrimerSensMinh (numeric(7,2), Null)
PrimerAdditionalInfo (varchar(60), Null)
PrimerSealant (varchar(20), Null)
BulletSealant (varchar(20), Null)
BulletMarking (varchar(20), Null)
CaseHeadMarking (varchar(20), Null)
dbo.Drawings
Columns
ProductCode (varchar(20), Not Null)
Version (int, Not Null)
DocumentType (tinyint, Not Null)
CompDrawing (image, Null)
BulletPath (image, Null)
BulletVelocity (image, Null)
dbo.QualityInspection
Columns
ProductCode (varchar(20), Not Null)
Version (int, Not Null)
DocumentType (tinyint, Not Null)
LottingInfo (varchar(200), Null)
dbo.TechnicalData
Columns
ProductCode (varchar(20), Not Null)
Version (int, Not Null)
DocumentType (tinyint, Not Null)
Identification (varchar(50), Null)
WaterTightness (varchar(200), Null)
ExtractionForcemin (int, Null)
ExtractionForcemax (int, Null)
CIPPressurePMax (int, Null)
CIPPressurePK (int, Null)
AverageMaxChamberPres (int, Null)
PressureMeasurementMethod (varchar(50), Null)
SingleMaxChamberPres (int, Null)
PressureTargetProduction (varchar(30), Null)
AverageMaxPortPress (int, Null)
SingleMaxPortPres (int, Null)
OverpressureTest (smallint, Null)
ActionTimeMax (int, Null)
OperatingTempMax (int, Null)
OperatingTempMin (int, Null)
BulletBC_G1 (numeric(4,3), Null)
BulletBC_G7 (numeric(4,3), Null)
BulletVelocity0 (int, Null)
BulletVelocity0Tolerance (int, Null)
BulletVel0STDDev (int, Null)
BulletVelocity2 (int, Null)
BulletVelocity2Tolerance (int, Null)
BulletVel2STDDev (int, Null)
BulletVelocity2Distance (int, Null)
BulletVelocitySeries (int, Null)
VelocityBarrelLength (int, Null)
BulletEnergy (int, Null)
BulletEnergy2 (int, Null)
BulletEnergy2Distance (int, Null)
BulletAccuracyMax1 (int, Null)
AccuracySpec1 (varchar(60), Null)
AccuracyDistance1 (int, Null)
BulletAccuracyMax2 (int, Null)
AccuracySpec2 (varchar(60), Null)
AccuracyDistance2 (int, Null)
AccuracyTarget (varchar(100), Null)
ArmourPiercingSpec1 (varchar(30), Null)
ArmourPiercingSpec2 (varchar(30), Null)
ArmourPiercingPlate (varchar(50), Null)
Trajectory (varchar(50), Null)
RiflingTwist (varchar(30), Null)
BulletIntegrity (varchar(100), Null)
Special1 (varchar(100), Null)
Special2 (varchar(100), Null)
FunctionRifles (varchar(100), Null)
BallisticData (varbinary(6000), Null)
BallisticDataInput (varchar(150), Null)
AccuracyBarrelMake (varchar(30), Null)
AccuracyBarrelLength (int, Null)
AccuracyBarrelGrooves (varchar(30), Null)
VelocityBarrelMake (varchar(30), Null)
VelocityBarrelGrooves (varchar(30), Null)
PressureBarrelMake (varchar(30), Null)
PressureBarrelLength (int, Null)
PressureBarrelGrooves (varchar(30), Null)

Liite 5. Järjestelmän ohjaustaulujen rakenne




dbo.FieldListContents
Columns
ListID (smallint, Not Null)
TypeID (tinyint, Not Null)
TypeDescription (varchar(75), Null)
PackingDataFormat (varchar(250), Null)
FieldOrder (int, Not Null)
dbo.FilePublishFormatting
Columns
DocumentType (int, Not Null)
Level (int, Not Null)
LevelStringFormat (varchar(750), Null)
LevelIsFile (bit, Null)
dbo.KaliiperienPMax
Columns
Caliber (varchar(255), Null)
PMaxSAAD (int, Null)
Pmaxbar (int, Null)
Aloituspaine (int, Null)
CIPMaxPressMPa (int, Null)
dbo.ListsToFields
Columns
ListID (smallint, Not Null)
FieldName (varchar(100), Not Null)
ListDescription (varchar(30), Null)
FormattedFieldName (varchar(150), Null)
TableName (varchar(150), Null)
dbo.PendingActions
Columns
ActionID (varchar(25), Not Null)
ProductCode (varchar(20), Not Null)
DocumentType (int, Not Null)
Version (int, Not Null)
Finished (bit, Null)
PendingFor (varchar(1), Not Null)
StartTime (datetime, Not Null)
FinishTime (datetime, Null)
SendingUser (varchar(100), Not Null)
RecievingUser (varchar(100), Not Null)
ActionMessage (varchar(4000), Null)
dbo.UserGroupDBManagementRights
Columns
UserGroupID (int, Not Null)
TableID (int, Not Null)
dbo.UserGroups
Columns
GroupID (smallint, Not Null)
GroupName (varchar(100), Not Null)
GroupDescription (varchar(450), Null)
dbo.UserGroupRights
Columns
UserGroupID (int, Not Null)
UserRightID (int, Not Null)
dbo.UserRightDefinitions
Columns
UserRightID (int, Not Null)
UserRightName (varchar(100), Not Null)
UserRightDescription (varchar(500), Null)
dbo.Users
Columns
UserID (int, Not Null)
WinUserAccount (varchar(100), Not Null)
FullName (varchar(150), Not Null)
Email (varchar(150), Not Null)
UserLetters (varchar(3), Not Null)
UserGroupID (smallint, Not Null)
Active (bit, Not Null)
dbo.TableColumns
Columns
ColumnsVersion (int, Null)
DBTable (varchar(100), Null)
ColOrder (int, Null)
FieldName (varchar(100), Null)
dbo.TechnicalDataFormating
Columns
FieldName (varchar(100), Not Null)
HeaderFormat (varchar(100), Null)
ValueFormat (varchar(100), Null)
PrintOrder (int, Null)