Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Sähköisen liiketoiminnan järjestelmät

2011

Pavel Gavrilov

VARAOSALUETTELON SUUNNITTELU JA TOTEUTUS ASP.NET -YMPÄRISTÖÖN

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Sähköisen liiketoiminnan järjestelmät

Marraskuu 2011 | Sivumäärä: 39

Ohjaaja: Päivi Nygren

Pavel Gavrilov

VARAOSALUETTELON SUUNNITTELU JA TOTEUTUS ASP.NET-YMPÄRISTÖÖN

Tämä opinnäytetyö kertoo Saides Engineering Oy:n pyynnöstä tehdystä projektista, jonka tuloksena oli varaosaluettelon suunnittelu, toteutus ja käyttöönotto ASP.NET-ympäristöön.

Saides Engineering Oy on palveleva insinööritoimisto ja tuotannollisten yritysten monipuolinen yhteistyökumppani, jonka tuotteiden kokoluokka vaihtelee hienomekaniikasta konepajatuotteisiin. Insinööritoimiston toteuttamien tuotteiden tiedot ylläpidetään MS SQL -tietokantapalvelimen avulla.

Opinnäytetyössä kuvatun projektin tavoitteena oli toteuttaa Internet-verkossa toimiva ohjelmisto, joka ylläpitäisi Saides Engineering Oy:n tekemistä tuotteista ajantasaisen luettelon. Luettelon tietolähteenä oli käytettävä Saides Engineering Oy:n MS SQL-tietokantapalvelinta ja ohjelmakoodin kielenä Visual Basic .NET:iä. Ohjelmiston on toimittava sulavasti yrityksen palvelinympäristössä.

Projektia mallina käyttäen työ kertoo ohjelmistotuotannon vaiheista asiakkaan toiveesta valmiin ohjelmiston käyttöönottoon. Projekti on alusta loppuun saakka noudattanut sille valittua toteutussuunnitelmaa. Ohjelmointityökaluna on käytetty Microsoft Visual Studio 2010 Professionalia ja ohjelmointikielenä VB.NET:iä ja ASP.NET:iä.

Projektin lopputuloksena toteutettu ohjelmisto on hyväksytty ja otettu tuotantokäyttöön. Suurin osa asiakkaan toivomia toimintoja saatiin toteutettu projektille määrätyssä ajassa. Ohjelmiston käyttäjien palaute on ollut positiivista.

ASIASANAT:

ohjelmistotuotanto, järjestelmäkehitys, Microsoft SQL Server, Visual Basic .NET, ASP.NET

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology | e-Business System

November 2011 | Total number of pages: 39

Instructor: Päivi Nygren

Pavel Gavrilov

DESIGNING AND IMPLEMENTATION OF SPARE PARTS CATALOG IN ASP.NET ENVIRONMENT

This thesis describes the design and implementation project of spare parts catalog in ASP.NET environment done for Saides Engineering Oy.

Saides Engineering Oy is a customer-oriented engineering office and a versatile partner for manufacturing companies. The scale of their products ranges from precision mechanics to heavy machinery. Data about products done by the engineering office is stored on MS SQL-database server.

The purpose of the project described in this thesis was to implement software running over the Internet that maintains the catalog of products manufactured by Saides Engineering Oy. The catalog should be based on MS SQL- database server of Saides Engineering Oy. Software uses Visual Basic .NET programming language and should work smoothly in company software environment.

The thesis uses the project model to show all steps of software engineering from the customer wish to final product. The project followed chosen implementation plan from start to finish. Microsoft Visual Studio 2010 Professional was used as programming tool for ASP.NET environment development and programming language was VB.NET.

The result software of the project was taken into production use. Most of customer wished functions were reached in the project time. The feedback given on the final software was strongly positive.

KEYWORDS:

software engineering, system development, Microsoft SQL Server, Visual Basic .NET, ASP.NET

SISÄLTÖ

1 JOHDANTO **6**

2 OHJELMISTOTUOTANNON MENETELMISTÄ **7**

2.1 Elinkaarimallit yleisesti 7

2.2 Ketterät menetelmät 8

2.3 Feature Driven Development (FDD) 10

3 PROJEKTIN TAVOITTEET JA MÄÄRITTELY **12**

3.1 Lähtötilanne 12

3.2 Projektin tavoitteet 12

3.3 Projektin määrittely 13

4 SUUNNITTELU **14**

4.1 Suunnittelun vaiheet 14

4.2 Käyttöliittymän ensimmäinen versio 15

4.3 Toimintolistan kerääminen ja tietokantamallin suunnittelu 16

4.4 Projektin aikataulun suunnittelu 19

5 TOTEUTTAMINEN **20**

5.1 Ohjelmoinnin aloittaminen 20

5.2 Ensimmäinen versio ohjelmasta (0.1) 21

5.3 Toinen versio ohjelmasta, laajennettu (0.2) 23

5.4 Esittelykelpoinen versio ohjelmasta (0.5) 25

5.5 Lopullinen versio ohjelmasta (1.0) 28

6 KÄYTTÖÖNOTTO **31**

6.1 Käyttöönoton suunnittelu 31

6.2 Ohjelman turvallisuuden parantaminen 31

6.3 Tietokantojen päivityksen järjestäminen 32

6.4 Käyttäjien lisääminen 34

7 ARVIOINTI JA OHJELMAN JATKOKEHITYKSEN IDEOINTI **36**

7.1 Projektin arviointia 36

7.2 Ohjelman jatkokehityksen ideoita 37

KUVAT

Kuva 1. Vesiputousmalli 8
Kuva 2. Agile Model Driven Development 9
Kuva 3. FDD Initial Process 11
Kuva 4. Ensimmäinen visio ohjelmasta 15
Kuva 5. Tietokantarakenne 18
Kuva 6. Luotu SQL-näkymä MS SQL Server puolelle 22
Kuva 7. Ensimmäinen versio ohjelmasta 23
Kuva 8. Toinen versio ohjelmasta 25
Kuva 9. Kolmas versio ohjelmasta, esitetty tilaajalle 27
Kuva 10. Virheilmoitus. Tuotetta ei löydy. 29
Kuva 11. Lopullinen versio ohjelmasta 30
Kuva 12. IIS-virheilmoitus 32
Kuva 13. Tietokantojen päivittäminen Saides Oy:n palvelimelle 34

TAULUKOT

Taulukko 1. Toimintolista 17
Taulukko 2. Ohjelmaversiot selityksineen 20

1 JOHDANTO

Saides Engineering Oy on insinööritoimisto, joka suunnittelee yhteistyökumppanien pyynnöstä erilaisia tuotteita. Yrityksen erikoisosaamisen ala on hydraulisia järjestelmiä sisältävät koneet ja laitteet. Saides Engineering Oy ylläpitää tietoja valmistamistaan tuotteista MS SQL -tietokantapalvelimien avulla. Yrityksen toimitusjohtaja itse valmistaa sovelluksia Visual Basic for Application -ohjelmointikielellä tietokantadatan hallinnointia varten.

Insinööritoimiston käyttämät sovellukset toimivat vain lähiverkossa ja yritys on halunnut saada tuoteluettelon tuotteistaan, joka olisi käytössä Internet-verkon välityksellä. Sovelluksen on toimittava sulavasti yrityksen sovellusympäristössä ja ohjelmointikieleksi valittava Visual Basic for Application -ohjelmointikielen tyyppinen ohjelmointikieli.

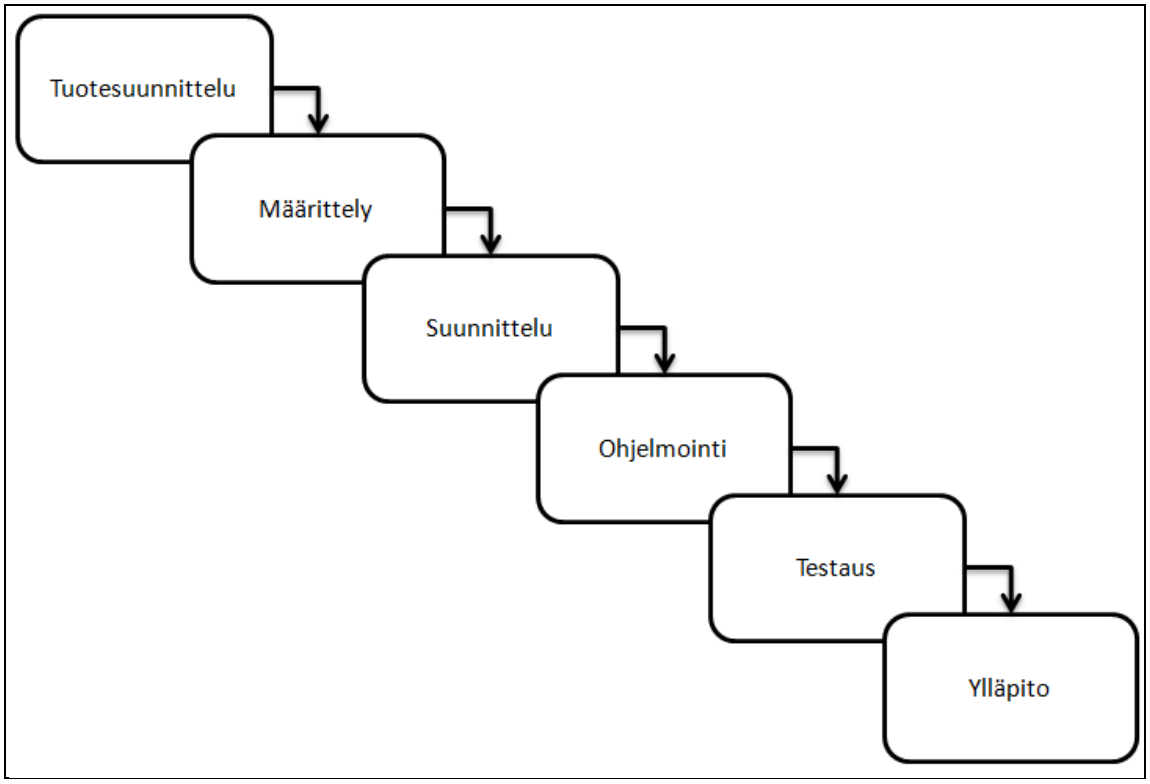
Opinnäytetyö kuvaa pyydetyn sovelluksen toteutuksen kaikki vaiheet asiakkaan toiveesta valmiin sovelluksen käyttöönottoon. Sovellus toteutettiin Visual Basic .NET-ohjelmointikielellä ASP.NET-ympäristöön. Työssä kuvataan ohjelmistotuotannon ketterien menetelmien mukaiset tyypilliset vaiheet ja Microsoftin ohjelmointiympäristössä tyypilliset ohjelmointi- ja käyttöönottomenetelmät.

2 OHJELMISTOTUOTANNON MENETELMISTÄ

2.1 Elinkaarimallit yleisesti

Ohjelmistotuotannon elinkaarimallilla yleensä tarkoitetaan standardisoitua menetelmää, joka pitää sisällään kaikki ohjelmistotuotannon vaiheet asiakkaan ideasta ohjelmiston käyttöönottoon. Elinkaarimallin rakenne auttaa hahmottamaan, missä järjestyksessä ohjelmiston tuotannossa edetään. Hyvin suunniteltu elinkaarimalli määrittelee ohjelmistotuotannon prosessit ja auttaa luomaan korkealaatuisen ohjelmiston systematisoidulla, tehokkaalla ja hallitulla tavalla. (Sommerville 2007, 11)

Elinkaarimallit voidaan jakaa lineaarisesti eteneviin, kuten esimerkiksi vesiputousmalli, RUP-malli, ja iteratiivisiin ja inkrementaalisiin, kuten spiraalimalli, Feature Driven Development, protoilumallit, Desingn-to-Schedule ym. ketterien menetelmien elinkaarimallit. Oikean elinkaarimallin valitseminen helpottaa projektin etenemistä ja seurantaa. Vesiputousmallin vaiheet kronologisessa järjestyksessä ovat: tuotesuunnittelu, vaatimusanalyysi, määrittely, suunnittelu, ohjelmointi, testaus, julkaisu ja julkaisun jälkeinen ylläpito. (Tian 2005, 44). Kuvassa 1 on esitetty variaatio perinteisestä vesiputousmallista.



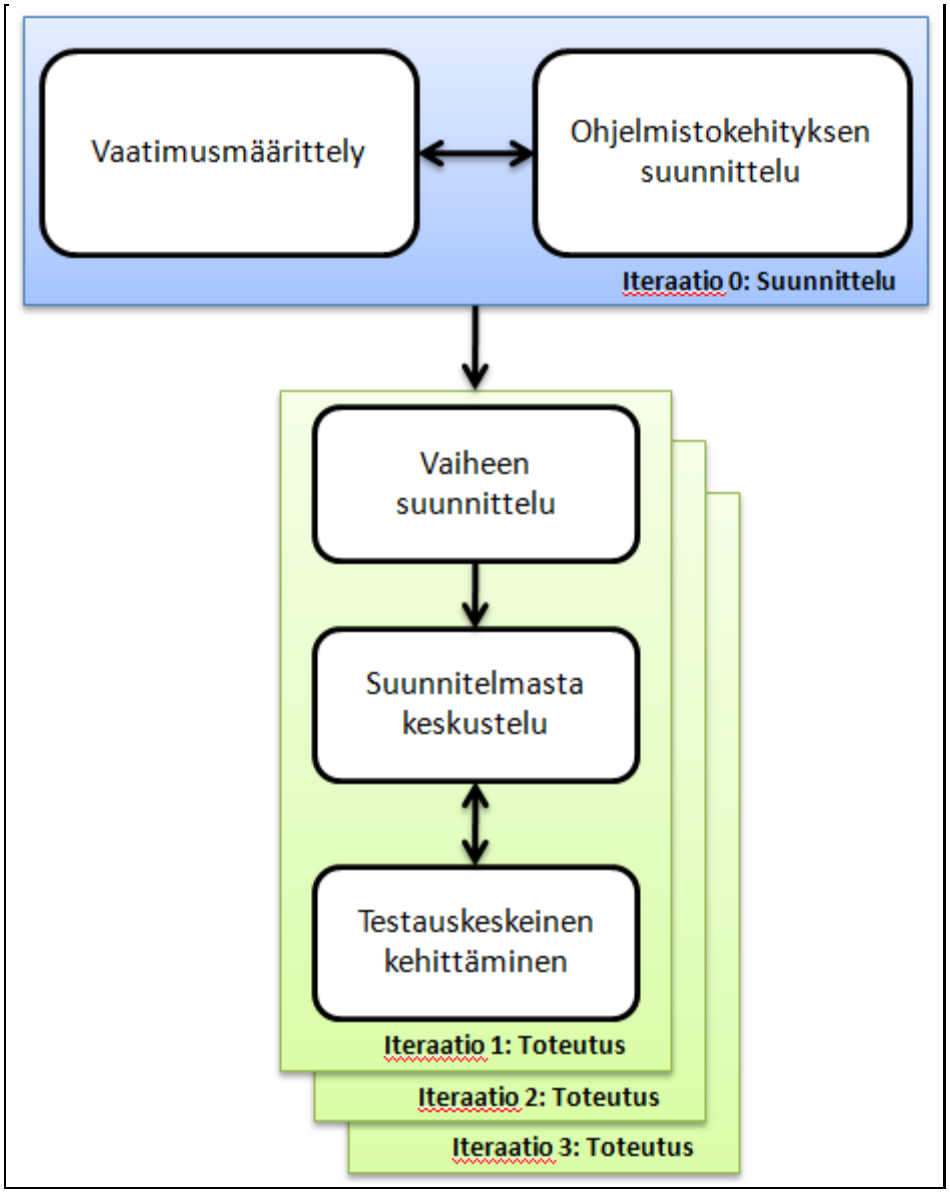
Kuva 1. Vesiputousmalli (Pressman 1994, 24)

Iteratiiviset ja inkrementaaliset elinkaarimallit yleisesti koostuvat useasta toistuvasta kierroksesta samoja vaiheita, kuten vesiputousmallissa. Jotta toimintavarmuus säilyisi joka iteraation jälkeen, jatkuva testaus on tärkeä osa näitä prosesseja. (Tian 2005, 46)

2.2 Ketterät menetelmät

Ketterien menetelmien pääperiaatteet ovat asiakastyytyväisyys, vaihtuvien vaatimusten hyväksyminen, useiden prototyyppimallien tuottaminen, yksinkertaisuus, toimiva ohjelmisto päämääränä, jatkuvan työtahdin ylläpito. Ketterät projektit vaativat liikemiesten ja kehittäjien kommunikointia, projektien ja työympäristön rakentamista motivoituneiden ihmisten ympärille, tiimin kasvokkain tapahtuvaa yhteistyötä, teknisen osaamisen ylläpitoa, itseorganisoituvien tiimien rakentamista ja tehokkuuden parantamisen ajattelua. (Agile Manifesto 2001)

Ketterien menetelmien mukaisten ohjelmistokehityksen tuotantomallien yhteiset piirteet ovat nopea iteraatiosykli ja asiakaspalautteiden ja -ehdotusten huomioiminen jokaisessa ohjelmiston kehitysvaiheessa. Kuva 2 esittää kaikkien ketterien elinkaarimallien tyypillisimmät vaiheet.



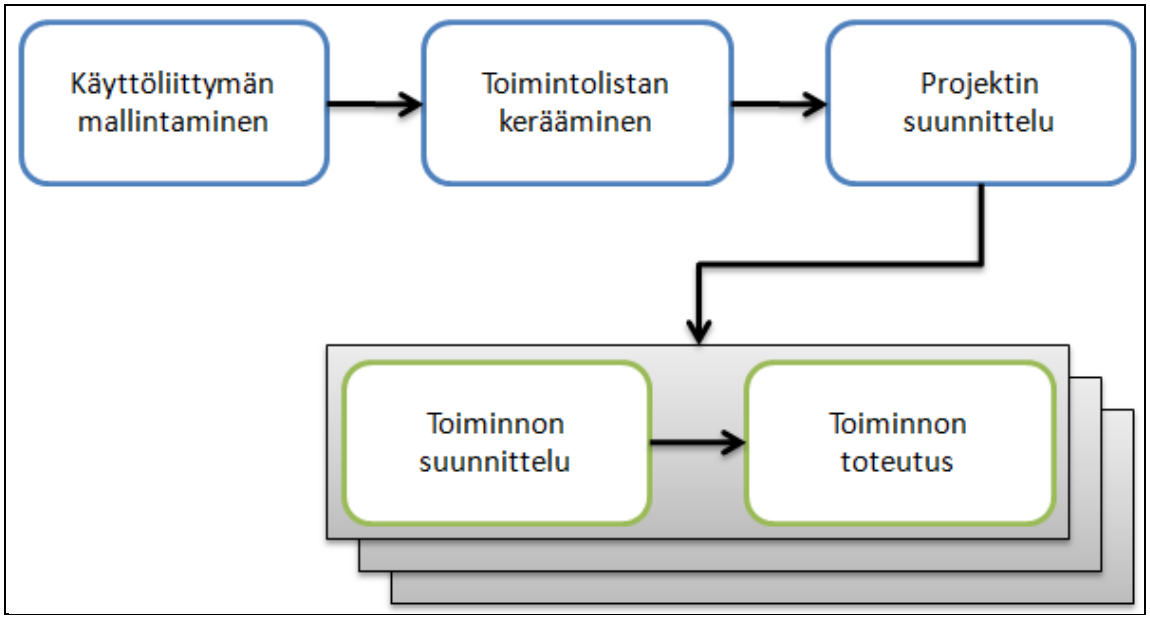
Kuva 2. Agile Model Driven Development (Amber 2003, AMDD)

Tässä projektissa ketterien menetelmien käytäntöjä on toteutettu, sillä projektin aikana on työskennelty tiiviissä yhteistyössä toimeksiantajan ja asiakkaan kanssa, julkaistu useat prototyyppimallit ohjelmasta sekä huomioitu myös myöhäisessä vaiheessa päivittyneet vaatimusmäärittelyt.

2.3 Feature Driven Development (FDD)

Feature Driven Development on ketterien menetelmien mukainen ohjelmistokehityksen elinkaarimalli, jonka ovat kehittäneet Jeff De Luca ja Peter Code. Sen pääpiirteitä ovat ominaisuuslistan laatiminen ja ominaisuuksien toteuttaminen yksi kerrallaan joka iteraatiokierroksella. (Khramtchenko 2004, 3)

FDD:n hyvänä piirteenä on tarkan toimintosuunnitelman ja ohjelman käyttöliittymän prototyyppiversion suunnittelu ennen varsinaista koodaamista. Samalla pystyy paremmin hahmottamaan, millaiset ohjelman ominaisuudet on mahdollista saada tietyssä ajassa. Myös etenemisen näkyvyys esimiehille, nopeat prototyyppiversiosyklit, ja helppo uusien toimintojen lisääminen toimintolistaan ovat Feature Driven Development elinkaarimallin etuja. (De Luca 2003, Issue 2)



Kuva 3. FDD Initial Process (Nebulon 2005)

Tässä opinnäytetyössä käsiteltävään projektiin FDD on valittu mentelmän yksinkertaisuuden ja läpinäkyvyyden takia. Nopeaa ensimmäistä prototyyppiversiota pidettiin tärkeänä mahdollisuutena. Myös se, että asiakkaalla ei ollut tarkkoja vaatimuksia ohjelman suhteen, mutta oli visio mahdollisista ominaisuuksista, vaikutti nimenomaan tämän kehitysmallin valintaan.

3 PROJEKTIN TAVOITTEET JA MÄÄRITTELY

3.1 Lähtötilanne

Saides Engineering Oy:n Salon yksiköllä on Microsoft SQL Server 2008 R2 -alustalla sijaitsevat tietokannat, joiden ylläpito ja raporttien tulostaminen on toteutettu lähiverkossa useimmiten joko suoraan tietokannan hallintajärjestelmän omilla työkaluilla ja menetelmillä tai Microsoft Office Access tietokantojen hallintaohjelmaan suunniteluilla työkaluilla ja ohjelmilla.

Tietokannat sisältävät muun muassa dataa yrityksen suunnittelemista tuotteista erilaisille yhteistyökumppaneille viidellä eri kielellä: suomeksi, englanniksi, ruotsiksi, saksaksi ja ranskaksi.

Microsoft Office Access -tietokantojen hallintaohjelmassa toimivat työkalut ja ohjelmat hyödyntävät toiminnassaan paljon Visual Basic for Applications (VBA) -ohjelmointikieltä, josta Saides Engineering Oy:n Salon yksikön toimitusjohtajalla ja projektin vetäjällä on monen vuoden vankka kokemus.

3.2 Projektin tavoitteet

Saides Engineering Oy:n Salon yksikkö tarvitsi Internet-verkossa toimivan sivuston, joka pystyisi tulostamaan asiakkaan näytölle palvelimen tietokannoista poimittua dataa yrityksen suunnittelemista tuotteista halutulla kielellä. Sen lisäksi sivustosta on pystyttävä lataamaan asiakkaan omalle koneelle tuotekuvauksen PDF-versio tuotelehtisestä samalla kielellä.

Sivuston siis piti toimia eräänlaisena hakukoneena tietokantojen dataa käyttäen. Asiakkaalla ei ollut valmista ratkaisua, miltä sivuston pitää näyttää ja mitkä kaikki ominaisuudet sen kuuluu sisältää, vaan oli hyvin ajateltu visio siitä, millaisia tietoja eri hakukriteereillä pitäisi löytyä.

3.3 Projektin määrittely

Projekti oli alusta saakka suunniteltu niin, että sitä voisi jatkossakin parantaa ja ylläpitää. Valmiin sivuston piti myös toimia sulavasti Saides Engineering Oy:n ohjelmistoympäristössä noudattaen jo aikaisemmin käyttöön otettuja käytäntöjä.

Sen ansiosta, että muut Saides Engineering Oy:n WWW-verkossa olevat sivut toimivat Internet Information Services -alustalla, projektin tuloksena syntyneen ohjelmiston piti myös pystyä toimimaan samalla alustalla. Sen johdosta ASP.NET -teknologian käyttö oli luonteva valinta jo määrittelyvaiheessa. ASP.NET -teknologia voi hyödyntää kolmea ohjelmointikieltä: C#, F# ja Visual Basic .NET. Koska jatkokehitysmahdollisuus huomioitiin jo määrittelyvaiheessa, ohjelmointikieleksi valittiin Visual Basic .NET. Kyseinen kieli on rakenteeltaan ja ulkoasultaan hyvin samantapainen kuin VBA, josta projektin vetäjällä on osaamista.

Vaikka ohjelman pitää pyöriä Internetissä, sen tulostamat tiedot pitää olla vain määriteltyjen henkilöiden ja organisaatioiden saatavilla. Tämän takia sivuston pitää olla suojattu käyttäjätunnus- ja salasanayhdistelmällä. Käytössä olevien järjestelmien ja mahdollisuuksien pohjalta katsottiin parhaaksi ottaa käyttöön Windows-käyttäjätunnuksella kirjautuminen.

Käytössä olevan datan laajuus on niin suuri, että sen tulostaminen rajaamatta tekee siitä käyttökelvottoman. Tämä ohjasi siihen, että sivuston pitää toimia joko hyvin tarkasti jäsenneltynä luettelona tai hakukonetyyppisesti, jossa hakukriteerit rajaavat tarkasti, mitkä tiedot tulevat näkyviin.

4 SUUNNITTELU

4.1 Suunnittelun vaiheet

Projektin suunnittelu on edennyt useassa vaiheessa, joista jokaisessa on käytetty siihen tilanteeseen parhaiten sopivaa menetelmä. Voidaan sanoa, että koko projektin aikana on käytetty Feature-Driven Development -kehitysmallia. Malli on valittu sen takia, että suunnittelun alkuvaiheessa ei tarvitse olla selvää kuvaa koko ohjelman toiminnasta ja toimintojen suunnittelu- ja toteutusvaiheet ovat keskenään iteratiivisia. Sen lisäksi tällainen toimintatapa pakottaa tiiviiseen yhteistyöhön ohjelmoijan ja asiakkaan välillä, joka oli molempien osapuolten mielestä positiivinen asia.

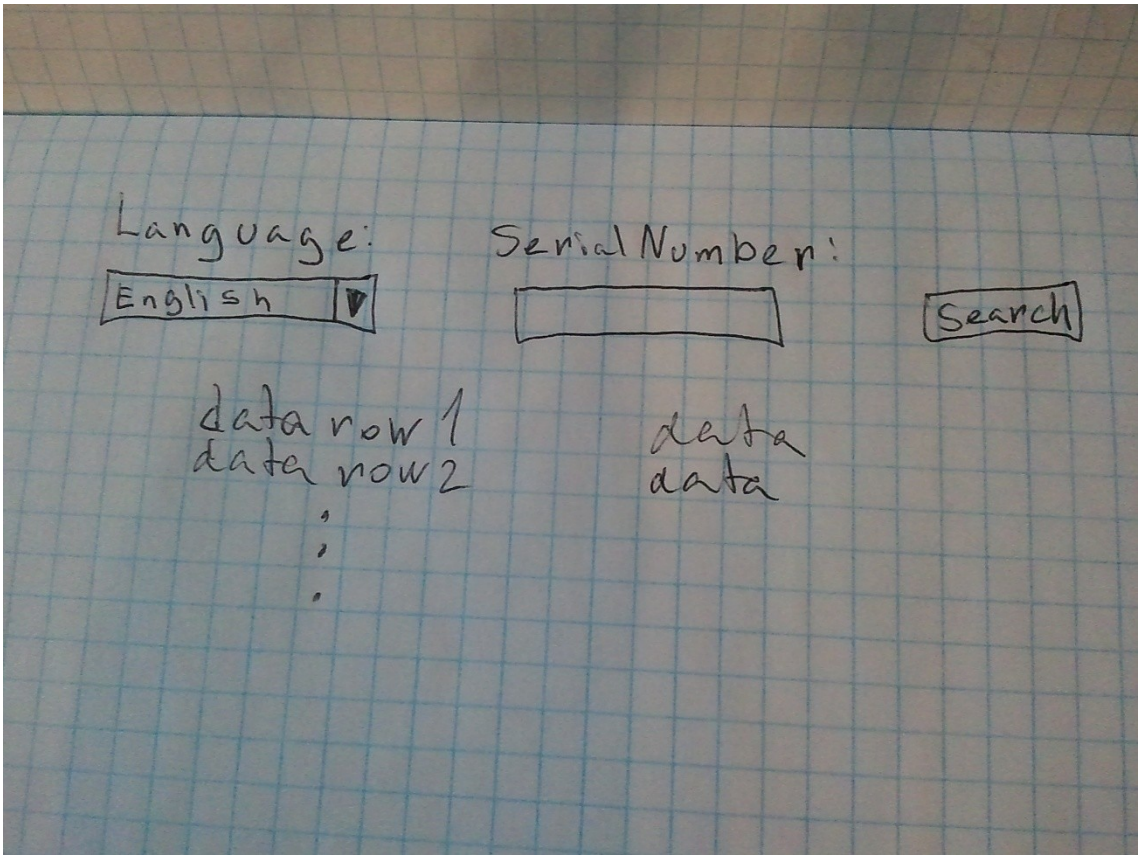
Käyttäen Feature-Driven Development -kehitysmallin termejä voin sanoa, että kolmen alkuvaiheen jälkeen olen käynyt läpi neljä kierrosta toimintojen suunnittelu- ja toteutusvaiheiden välistä siirtymistä sekä lopullisen ulkonäköversion toteutuksen jälkeen olen palannut karkean mallin kehittämisvaiheeseen ja toteuttanut saman kehitysmallin mukaisesti ohjelmaympäristön toimivuuden sekä ohjelmassa käytettyjen tietokantojen ylläpidon.

Suunnitteluosiossa kerron itse ohjelman FDD:n kolmesta ensimmäisestä vaiheista, joita ovat karkean mallin kehittäminen, toimintolistan kerääminen ja kehityksen suunnittelu. Toimintojen suunnittelun ja toteutuksen jätän lukuun Toteuttaminen, kun taas sen jälkeiset vaiheet esitän luvussa Käyttöönotto.

Sen takia, että toteutettu ohjelma käyttää tietokantoja ja tauluja, jotka ovat tuotannollisen yrityksen omistuksessa, kerron suunnittelu- ja toteutusvaiheissa käytetyistä menetelmistä käyttäen keksittyjä taulujen ja tietokantojen nimiä, sekä ohjelmakoodia, joka vastaa tyyliltään ja tasoltaan oikeassa ohjelmassa käytettyjä koodeja.

4.2 Käyttöliittymän ensimmäinen versio

Tässä vaiheessa yhdessä projektin vetäjän kanssa luotiin ensimmäinen versio käyttöliittymästä. Ensimmäisen vision ei tarvinnut olla tyyliltään oikeaoppinen, vaan sen tarkoitus oli päästä yhteisymmärrykseen siitä, mitä ylipäätänsä ollaan tekemässä. Sen takia käyttöliittymä oli pikaisesti suunniteltu ruutupaperille (kuva 4).



Kuva 4. Ensimmäinen visio ohjelmasta

Tällainen malli antoi hyvät eväät saada käsitys siitä, mihin suuntaan lähdetään ohjelmoimaan.

Samalla päädyimme tulokseen, että Saides Oy:n on hankittava ohjelma Microsoft Visual Studio 2010 Professional, jotta päästäisiin tekemään ohjelmistoa, jossa on .NET Framework 4.0 -tuki.

Ohjelman saavuttua asensin sen ja rupesin itsenäisesti tutustumaan sen ominaisuuksiin sekä katsomaan ASPX-malliohjelmia ymmärtääkseni, millä tavalla yhdistelmä ASP-koodi ja VB.NET toimii. Samalla olen hankkinut Internet-hakukoneiden avulla ilmaiseksi ladattavia ohjelmakoodeja, joita voisin hyödyntää työssäni. Sen lisäksi olen tutustunut kahdeksaan alan kirjaan, joista olen valinnut kolme, jotka mielestäni parhaiten sopivat työni käsikirjoiksi.

4.3 Toimintolistan kerääminen ja tietokantamallin suunnittelu

Tutustuttuani taustamateriaaliin aloin tutkia työympäristöni ja keräämään kaikki teoreettisesti mahdolliset toiminnot, jotka voitaisiin toteuttaa projektin sisällä. Toimintolistaan pääsivät myös sellaiset ominaisuudet, joiden toteuttaminen määrätyssä ajassa jo projektin alussa näytti epätodennäköiseltä. Taulukossa 1 on esiteltynä kaikki mahdolliset toiminnot, joista osa on toteutettu projektin aikana, osa jätetty kokonaan toteuttamatta ja osa suunniteltu projektin jatkokehitykseksi.

Toiminto	Toteutettiin	Ei toteutettu
Luettelo kaikista tuotteista numerojärjestyksessä tai muuten järkevästi jäsenneltynä.		✗
Tuotteen haku tuotenumeron mukaan. Tuotetiedot näkyvissä samalla sivulla.	✓	
Tuotteen osien tietoihin pääsee klikkaamalla näkyvissä olevaa tuotetta.	✓	
Tuotteesta voi ladata omalle koneelle kuvallisen PDF-version tuotteentiedoista.	✓	
Pitää olla kielenvalintamahdollisuus, PDF-tiedostot pätää olla myös kielikohtaiset.	✓	
PDF-tiedostot muodostetaan suoraan tietokannasta online-tilassa.		✗
Tuotteen haku tuotenimen mukaan.		✗
Nimellisessä haussa tulostettava data ryhmiteltynä laajuutensa vuoksi.		✗
Kännykkäversio sivustosta. *		
Tuote voidaan tilata suoraan luettelosta tai hakutuloksista. Online verkkokauppa. *		

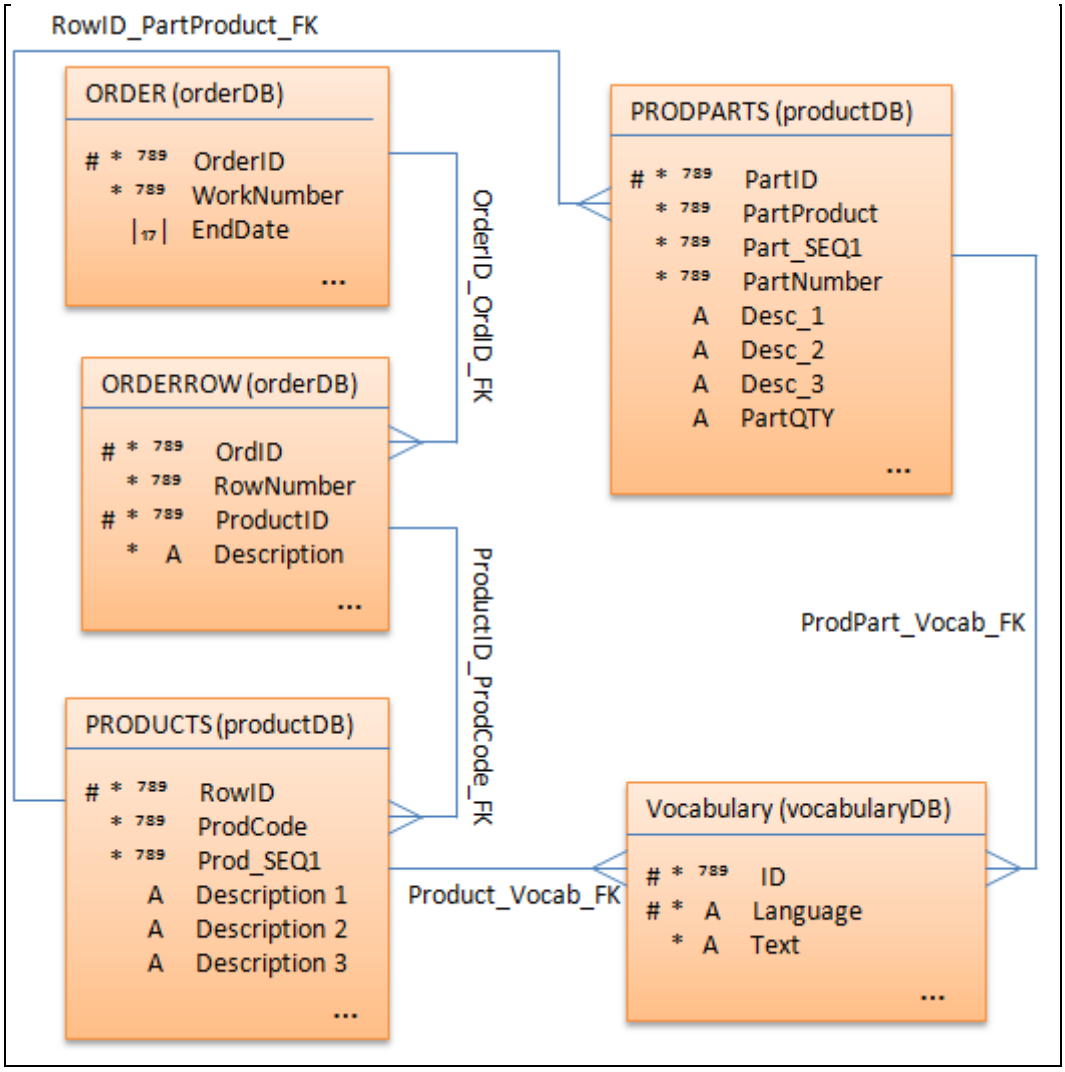
*) ei toteuta ominaisuuden laajuuden ja projektin ajan rajallisuuden takia

Taulukko 1. Toimintolista

Kuten näkyy taulukosta 1, toimintolistassa on paljon toisistaan erottuvia toimintoja, joista osa (esim. kaksi viimeistä) vaatisi oman erillisen ja laajahkon projektin. Tällainen lähestymistapa antoi ymmärryksen siitä, että ohjelman kehittäminen ei lopu projektin päätyttyä, vaan ollaan aloittamassa jatkuvasti kehittyvää ohjelmistoa. Se osaltaan korosti koodin selkeyden ja raportoinnin tärkeyden.

Sen lisäksi olen tutustunut tietokantoihin, joiden datan pohjalle minun piti luoda ohjelma. Kuvassa 5 on esitelty tietokantarakenne, joka laajuudeltaan ja yhteyksiltään vastaa oikeassa ohjelmassa hyödynnettyjä tietokantarakennetta. Kuten kuvassa näkyy, taulut sijaitsevat kolmessa eri tietokannassa. Tällainen jaottelu lienee syntynyt siitä, että alun perin tietokannat sisälsivät eri yrityksien dataa,

mutta yrityksien yhteistyön ansiosta tietokantojen datojen välille syntyi yhteyksiä. Esille on tuotu vain ne taulut ja kentät,joita käytettiin tässä projektissa kehitetyssä ohjelmistossa. Taulujen ja kenttien nimet on muutettu tietoturvallisista syistä. Hakukentäksi, jonka perusteella haut on suoritettava, valittiin taulun ORDER kenttä WorkNumber.



Kuva 5. Tietokantarakenne

4.4 Projektin aikataulun suunnittelu

Tässä vaiheessa sovittiin toimeksiantajan kanssa, millä tavalla projekti etenee, ja missä järjestyksessä toteutetaan erilaiset komponentit ja toiminnot. Ensimmäiseksi on laadittava sellainen ohjelman versio, jota voidaan näyttää lopullisille käyttäjille. Ensimmäinen ohjelman versio on ajettava ylös käytössä olevalle palvelimelle niin, että se olisi testitunnuksen kautta käytettävissä myös sisäisen verkon ulkopuolelta eli Internetistä. Sen jälkeen, kun saadaan palautetta loppukäyttäjiltä, kehitetään sellainen versio, joka on loppukäyttäjien mielestä hyvä ja toimiva. Tässä vaiheessa lisätään ohjelmaan kaikki ominaisuudet, jotka ovat loppukäyttäjän mielestä pakollisia. Kun lopullinen versio ohjelmasta on valmis, järjestetään tietokannan datan automatisoitu päivitys. Tämän jälkeen poistetaan testikäyttäjä ohjelmasta ja lisätään oikeat käyttäjät ohjelmaan. Jos sen jälkeen jää vielä aikaa ohjelman kehitykseen, ruvetaan suunnittelemaan lisäominaisuuksia ohjelmaan. Ensimmäisen loppukäyttäjien hyväksymän ohjelmaversion ylösajon viimeiseksi päivämääräksi määrättiin 31. elokuuta 2011, eli kolme kuukautta projektin alkamisesta.

5 TOTEUTTAMINEN

Tässä osiossa kerron ohjelman eri versioiden kehittämisestä ensimmäisestä versiosta, jota näytin toimeksiantajalleni siihen lopputulokseen, jota lopulliset käyttäjät ovat hyväksyneet. Sen jälkeiset toimenpiteet, kuten tietokantojen datan päivittäminen ja loppukäyttäjien lisääminen ohjelmaan esitän osiossa Käyttöönotto. Taulukossa 2 on esitetty kaikki ohjelmaversiot selityksineen.

Käyttöliittymän versio	Ominaisuuksien selitys
Versio 0.1	Ensimmäinen versio, jossa on toteutettu haku tuotenumeron mukaan. Hakutuloksissa mukana vain tuotteen tärkeimmät tiedot. Viisi hakukieltä.
Versio 0.2	Laajennettu versio edellisestä. Tuotteenosien tiedot myös näkyvissä hakutuloksissa. Automaattinen päivitys ominaisuus.
Versio 0.5	Lisätty kielikohtaiset linkit PDF-tuotelehtisiin. Parempi turvallisuus rakenne koko ohjelmassa. Virheilmoitukset muutettu. Ohjelma esitetty asiakkaalle.
Versio 1.0	Lopullinen versio. Otettu käyttöön. Toteutettu asiakkaan pyytämiä muutoksia ohjelmaan. Englanti on osana muita hakukieliä. Tuotteen nimi ja valmistusvuosi näkyvät ennen hakutuloksia. Asiakkaan logo ja otsikko näkyvissä.

Taulukko 2. Ohjelmaversiot selityksineen

5.1 Ohjelmoinnin aloittaminen

Alussa oli tärkeää luoda toimiva yhteys tietokantaan ja saada näytölle ohjelman avulla jotain dataa siitä, jotta voitaisiin todeta ratkaisun ylipäätänsä toimivan. Ensimmäiseksi loin tietokantayhteyden Visual Studion oman wizardin avulla. Se

antoi mahdollisuuden luoda automaattisesti päivittyvä kenttä, johon dataksi valitsin Products-taulun kentän RowID. Se tulosti kaikki tietueet samalle sivulle, mikä ei tieteenkään voinut olla toimiva ratkaisu. Toisaalta, se näytti, että yhteys tietokantaan on muodostettavissa, mutta sen muokkaaminen onnistui vain sillä samalla wizardilla. Siihen siis piti etsiä parempi ratkaisu.

Rupesin etsimään parempia ratkaisuja käsikirjoistani. Tietokantayhteyden muodostamiseksi on esitetty kaksi hyvää ratkaisua, ensimmäinen on laittaa se erilliseen web.config tiedostoon (Spaanjaars 2010, 445) toinen on laittaa suoraan VB.NET ohjelmaosioon (MacDonald 2010, 463). Valitsin näistä jälkimmäisen, koska se antoi paremmat mahdollisuudet nopeasti muokata connectionStringiä ohjelmointivaiheessa. Suunnittelin kuitenkin siirrettäväni yhteysrivin siirtämistä web.config -tiedostoon silloin, kun totean ohjelman muodostavan tietokantayhteyden varmatoimisesti, koska tämä tapa antaa paremmat jälkimuokkausmahdollisuudet.

Tietokannasta saadun datan tallentamiseksi kahdesta hyvästä vaihtoehdosta, DataReader (MacDonald ym. 2010, 287–289) ja DataSet (MacDonald 2010, 486-487) valitsin jälkimmäisen, koska se antaa paremmat mahdollisuudet käsitellä saatua dataa, samalla kokeilin valita datan tulostustavaksi DataList, joka siinä vaiheessa näytti hyvin toimivan DataSetin yhteydessä.

Sen takia, että haettava data on hajautettu monen eri taulun ja tietokannan välille, päädyin siihen ratkaisuun, että ensin muodostan tietokantahallinnon puolella näkymät, jotka kasaavat mahdolliset tulostuskentät yhdeksi näkymäksi ja sitten ohjelman puolella rajaan näkymästä tulostettavat hakukriteerien avulla.

5.2 Ensimmäinen versio ohjelmasta (0.1)

Edellisen vaiheen tulosten perusteella aloitin näkymien muodostamisella tietokantahallinnon puolelle käyttäen SQL Server Management Studio -ohjelmaa. Loin näkymät productDB -tietokantaan, koska yhteys siihen on todettu toimivaksi ja jotta kaikki näkymät olisivat samassa tietokannassa. Muodostin kaksi kyselyä. Ensimmäinen niistä toi esille kaikki Vocabulary-taulun kentät ja toimi niin

kuin sen kopiona productDB:n puolella. Toinen näkymä kasasi yhteen taulut Order, OrderRow ja Products ja käytti ensimmäisen näkymän. Kyselyn monimutkaisuuden takia esittelen sen kuvassa 6, jotta syntyisi parempi kuva siitä.

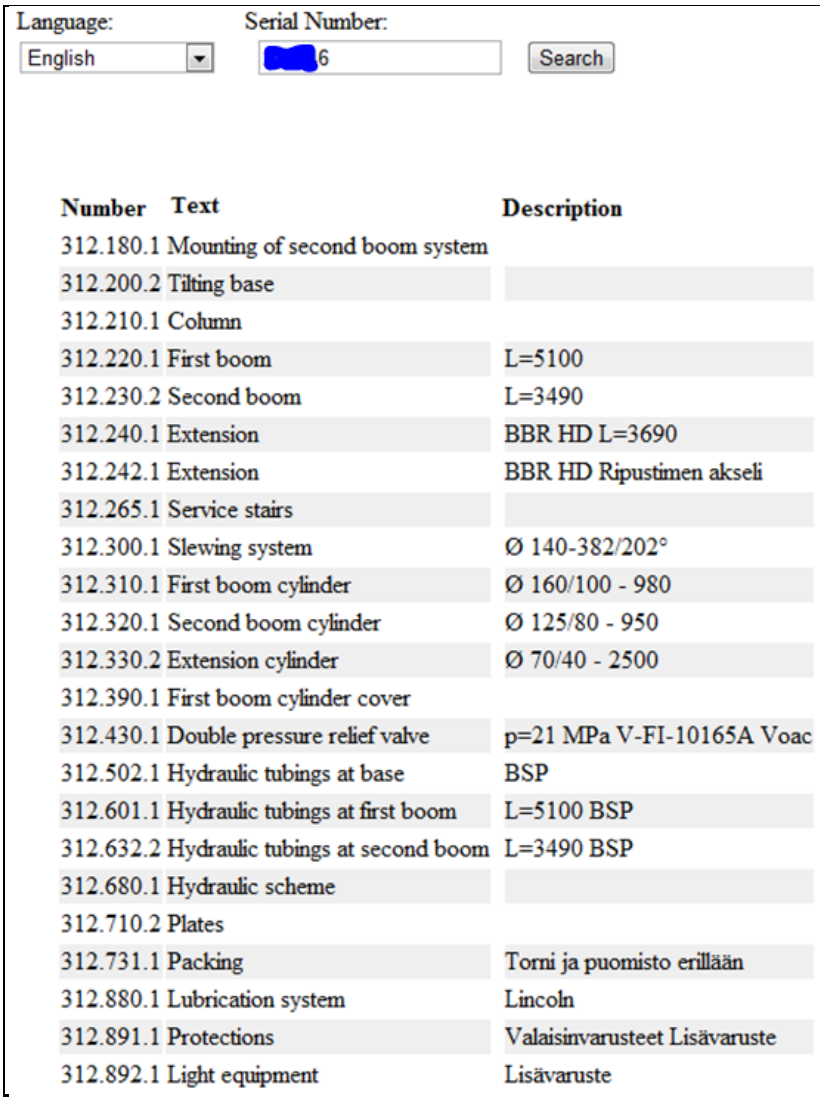
```
SELECT    productDB.dbo.Products.RowID, productDB.dbo.Vocabulary.Text,
          productDB.dbo.Products.[Description 1],
          productDB.dbo.Products.[Description 2],
          productDB.dbo.Products.[Description 3],
          productDB.dbo.Vocabulary.Language,
          OrderDB.dbo.Order.WorkNumber

FROM      OrderDB.dbo.Order INNER JOIN
          OrderDB.dbo.OrderRow ON OrderDB.dbo.OrderRow.OrdID =
          OrderDB.dbo.Order.OrderID INNER JOIN
          productDB.dbo.Products ON productDB.dbo.Products.ProdCode =
          OrderDB.dbo.OrderRow.ProductID INNER JOIN
          productDB.dbo.Vocabulary ON productDB.dbo.Vocabulary.ID =
          productDB.dbo.Products.Prod_SEQ1
```

Kuva 6. Luotu SQL-näkymä MS SQL Server puolelle

Tämän jälkeen laitoin ohjelmakoodin VB.NET-osioon muuttujan, johon tallensin tietokantayhteyden yhteysmerkkijonon (connectionString). Yhteys tässä vaiheessa muodostui suoralla SQL-tunnuksella ja salasanalla. VB.NET-ohjelmakoodiin tein myös oman sql-kyselyn, joka käytti pohjana yllä mainittuja näkymiä ja ASP.NET puolelle luotuja kenttiä ”Language” ja ”SerialNumber”.

Näin pääsin ensimmäiseen ohjelmaversioon. Se käynnisti haun ainoastaan, kun painoi ”Search” -nappulaa. Sen jälkeen kenttien ”Language”- ja ”SerialNumber”-arvot lähtivät Visual Studion virtuaaliselle serverille. Niiden perusteella muodostui SQL-kysely, joka tallentui DataSet:iin. Viimeinen puolestaan tulosti arvot DataList-kenttiin, joka lähetti ne takaisin ASP.NET-sivulle. Kuvassa 7 näkyy ohjel-

man ensimmäisen version yksi hakutulos. Kuvan avulla voi hahmotta paremmin ohjelman toimintaa.



Number	Text	Description
312.180.1	Mounting of second boom system	
312.200.2	Tilting base	
312.210.1	Column	
312.220.1	First boom	L=5100
312.230.2	Second boom	L=3490
312.240.1	Extension	BBR HD L=3690
312.242.1	Extension	BBR HD Ripustimen akseli
312.265.1	Service stairs	
312.300.1	Slewing system	Ø 140-382/202°
312.310.1	First boom cylinder	Ø 160/100 - 980
312.320.1	Second boom cylinder	Ø 125/80 - 950
312.330.2	Extension cylinder	Ø 70/40 - 2500
312.390.1	First boom cylinder cover	
312.430.1	Double pressure relief valve	p=21 MPa V-FI-10165A Voac
312.502.1	Hydraulic tubings at base	BSP
312.601.1	Hydraulic tubings at first boom	L=5100 BSP
312.632.2	Hydraulic tubings at second boom	L=3490 BSP
312.680.1	Hydraulic scheme	
312.710.2	Plates	
312.731.1	Packing	Torni ja puomisto erillään
312.880.1	Lubrication system	Lincoln
312.891.1	Protections	Valaisinvarusteet Lisävaruste
312.892.1	Light equipment	Lisävaruste

Kuva 7. Ensimmäinen versio ohjelmasta

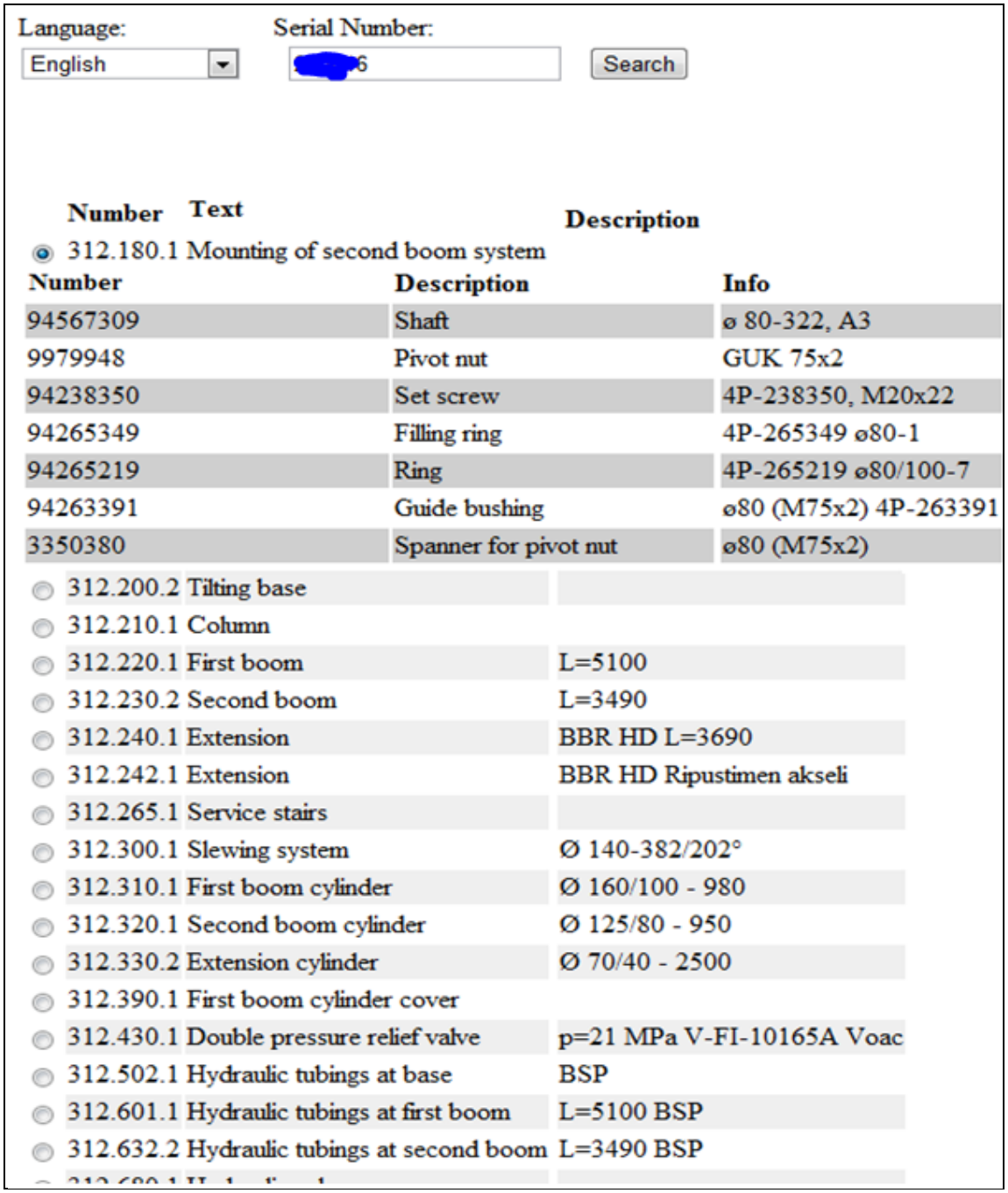
5.3 Toinen versio ohjelmasta, laajennettu (0.2)

Esitin ensimmäisen ohjelmaversion projektin vetäjälle ja hän totesi sen hyväksi ratkaisuksi. Seuraavana askeleena olisi lisätä ohjelmaan dataa taulusta ”Prod-Parts”. Projektin vetäjä ehdotti, että jokaisen moduulirivin (tulostetut rivit kuvassa 7.) edessä olisi ”+”-merkki linkkinä ja kun painaa plussaa, se muuttuu ”-”-merkiksi ja näytölle tulostuisivat tarvittavat tiedot.

Tein serverin puolelle taas uuden näkymän, joka oli hyvin kuvassa 6 näkymän tapainen, mutta kasasi data ”ProdParts”-taulun ympärille järkevään muotoon. Olen tutkinut mahdollisuuksia toteuttaa ”+”-merkki ratkaisua ja totesin kokeilun kautta, että omatekoiset JavaScript- ja VB.NET-funktiot eivät toimi DataList ja sen tapaisten rakenteiden sisällä. Sen lisäksi taulun ”ProdParts” dataa piti tuoda vain valitun rivin osalta, eli niitä ei saanut olla valmiina sivustolla piilotetussa muodossa. Näin siksi, että ohjelmasta olisi voinut tulla liian hidas, jos se hakisi paljon ylimääräistä dataa, jota ei käytetty juuri sillä hetkellä.

Jouduin siis etsimään jotain muuta ratkaisua tähän ohjelman toimintoon. Lukuisten kokeiluiden jälkeen päädyin RadioButtonList rakenteeseen (MacDonald 2010, 197-198) DataList-rakenteen sijasta. Se antoi mahdollisuuden seuraavalla funktiokierroksella ”napata” valittu rivi. Tein funktion, joka haki valitun rivin dataa ”ProdParts”-taulusta ja tulosti sitä suoraan sen rivin alle. Huono puoli oli siinä, että ohjelma teki uuden pääfunktiokierroksen vasta ”Search”-nappulan painalluksesta, mikä ei tieteenkään ollut hyvä käyttäjäkokemusta ajatellen. Löysin siihenkin nopeasti hyvän ratkaisun. Käytin AutoPostBack-ominaisuutta (MacDonald 2010, 192), joka automaattisesti luo käyttäjälle esitettävälle sivulle _doPostBack JavaScript-funktion. Funktio toimittaa serverin puolelle tiedot siitä, mikä kenttä on muuttunut, ja mikä on sen uusi arvo VB.NET-muuttujan muodossa. Nyt uusi pääfunktiokierros käynnistyi saman tien, kun valitsi radiobuttonin jonkin hakutulosrivin kohdalla. Pääfunktio vuorollaan toimitti uudelle funktiolle kaikki tarvittavat tiedot.

Sama AutoPostBack-ominaisuus on käytettävissä myös muissa rakenteissa. Päätin lisätä sen kielivalintakenttään ja hakukenttään. Näin sain tehtyä ohjelman, joka päivittää esitettävät tiedot heti, kun valitsee jonkun tulostusrivin tai vaihtaa kielen tai painaa ”Enter”-näppäintä. Lisäsin myös funktion, joka tarkistaa, ettei tyhjällä hakukentällä käynnistetä hakua. Kuva 8 esittää tuloksen yhdestä hausta uudella ohjelmaominaisuudella.



Kuva 8. Toinen versio ohjelmasta

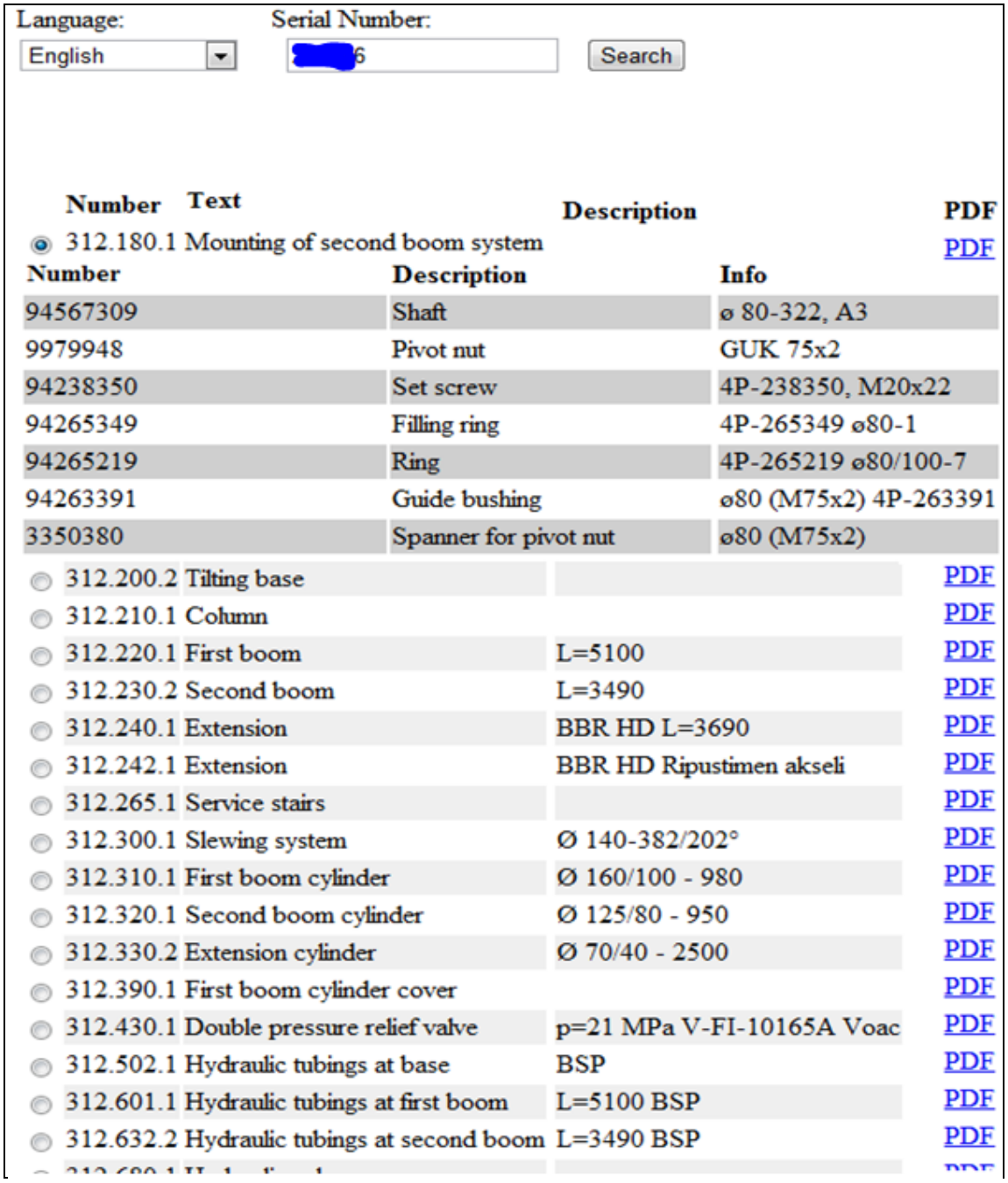
5.4 Esittelykelpoinen versio ohjelmasta (0.5)

Kun edellinen versio ohjelmasta oli esitetty projektin vetäjälle ja hyväksytty, jatkoin kehittämistä. Nyt oli vuorossaan PDF-tuotelehtisten tulostaminen jokaisesta näytöllä näkyvästä rivistä. PDF-tiedostot sijaitsivat serverillä jaettuna alikan-

sioihin sarjanumeron mukaisesti, ne ovat myös kielikohtaiset. Jokaista näkyvässä olevaa riviä varten ei välttämättä ole vielä valmista tuotelehtistä.

Tein serverille IIS:n kautta virtuaalikansion, joka linkittyi tuotelehtisten pääkansioon. Kehitin VB.NET-funktion, joka kielitietoja ja sarjanumeroa käyttäen haki serverin fyysisestä kansiosta mahdollisen PDF-tiedoston nimen ja jos löysi, teki linkin tiedostoa vastaavaan virtuaalikansioon. Jos tiedostoa ei jostain syystä löytynyt, funktio tulosti PDF-linkin sijasta tekstin ”file not found”.

Edellä mainitun funktion lisäksi paransin ohjelman turvallisuutta. Laitoin jokaisen VB.NET-funktion ja tietokantayhteyksien kohdalla funktion osat Try-Catch-Finally-End Try -lauseiden (MacDonald 2010, 211) sisään. Näin vaikka jokin ohjelman osa toimisi väärin, tai tietokantayhteydessä tulisi ongelmia, näytölle ei tulostuisi ohjelmakoodia, vaan määrätty virheilmoitus. Kuvassa 9 näkyy, miltä ohjelma näytti tässä vaiheessa.



Kuva 9. Kolmas versio ohjelmasta, esitetty tilaajalle

Projektin vetäjä oli tyytyväinen tulokseen ja sain luvan siirtää toimivan ohjelman tuotantopalvelimelle ja näin myös toimimaan Internetissä. Ainoana pyyntönä oli lisätä asiakkaan, Mesera Oy:n, logo ja ”Spare Parts Catalog”-teksti. Muutin tietokantayhteyden muodostamisen Windows-tunnuksen kautta toimivaksi.

Asensin sivuston toimimaan palvelimella niin, että vain ne Windows-tunnukset, joille on määrätty oikeus, pääsivät käyttämään ohjelmaa. Tein testitunnuksen, jonka tarkoituksena oli esittää ohjelman toimintaa asiakkaalle. Tallensin oikeutetut tunnukset web.config -tiedostoon (MacDonalds 2010, 643). Samoille tunnuksille annoin oikeuden tuotelehtisten kansion lukemiseen. Ohjelma oli valmis esitettäväksi toimeksiantajalle.

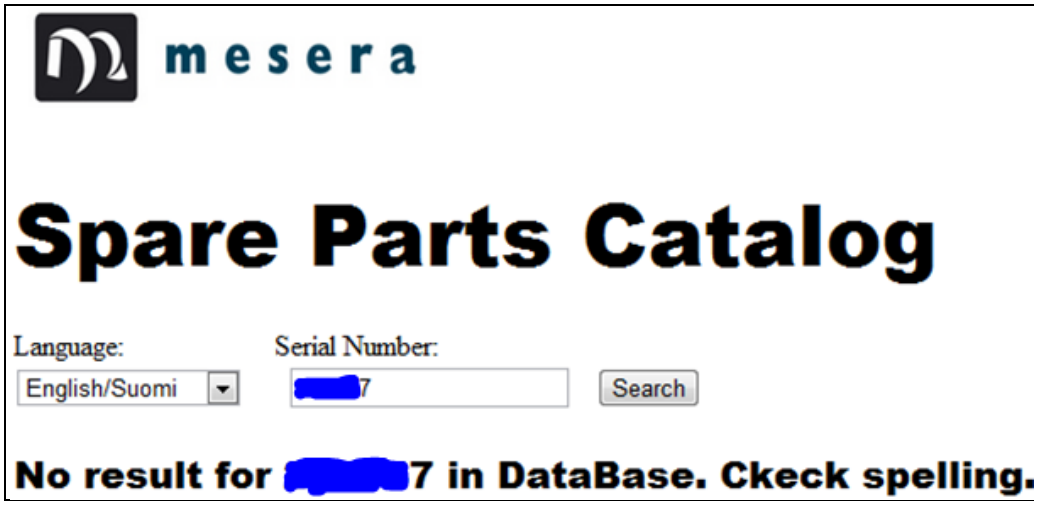
5.5 Lopullinen versio ohjelmasta (1.0)

Saimme toimeksiantajan asiakkaalta palautteen, joka oli rohkaisevasti positiivinen. Asiakas oli tyytyväinen ohjelman toimivuuteen ja hänen mielestään ohjelma vastasi hyvin odotuksia. Asiakas halusi kuitenkin muutamia muutoksia ohjelman ulkonäköön. Ensimmäisenä hän halusi ennen hakutulosten tulostamista rivin, jossa lukisi koneen nimi ja valmistuspäivämäärä, jonka osat on saatu hakutulokseen, jotta syntyisi varmuus siitä, että on haettu oikeaa tilausnumeroa. Sen lisäksi tulisi tehdä muutoksia kenttien nimityksiin ja lisätä muutama, asiakkaan mielestä olennainen kenttä.

Sen ohella Saides Oy on päivittänyt serveriään ja uudistuksen myötä oikeudet web-ohjelmiin on jaettava Active Directoryn kautta. Myös PDF-tuotelehtiset päivitettiin sellaiseen muotoon, jossa ei ollut erillistä englannin kieltä, vaan englanninkieliset nimitykset olivat osana muita kieliversioita. Ohjelman oli myös noudatettavaa uutta PDF-tuotelehtisten ulkonäköä.

Tein pyydettyjä muutoksia ohjelmaan. Jouduin muuttamaan kokonaan PDF-linkki funktiota, sekä ottamaan käyttöön vielä yhden näkymän SQL-Serverin puolella, jotta PDF-linkit edellen toimisivat ja englanninkieliset nimitykset tulostuisivat aina muiden kieliversioiden mukana. Tein myös funktion, joka tulostaisi koneen nimi ja käyttöönottopäivämäärän tai virheilmoituksen, jos konetta ei löydy tietokannasta haetulla työnumerolla. Funktiota varten tein myös uuden näkymän SQL-Serveriin. Sen lisäksi siirsin tietokantayhteysarvot web.config -tiedostoon, jotta niitä olisi helppo muuttaa käyttöönoton jälkeenkin.

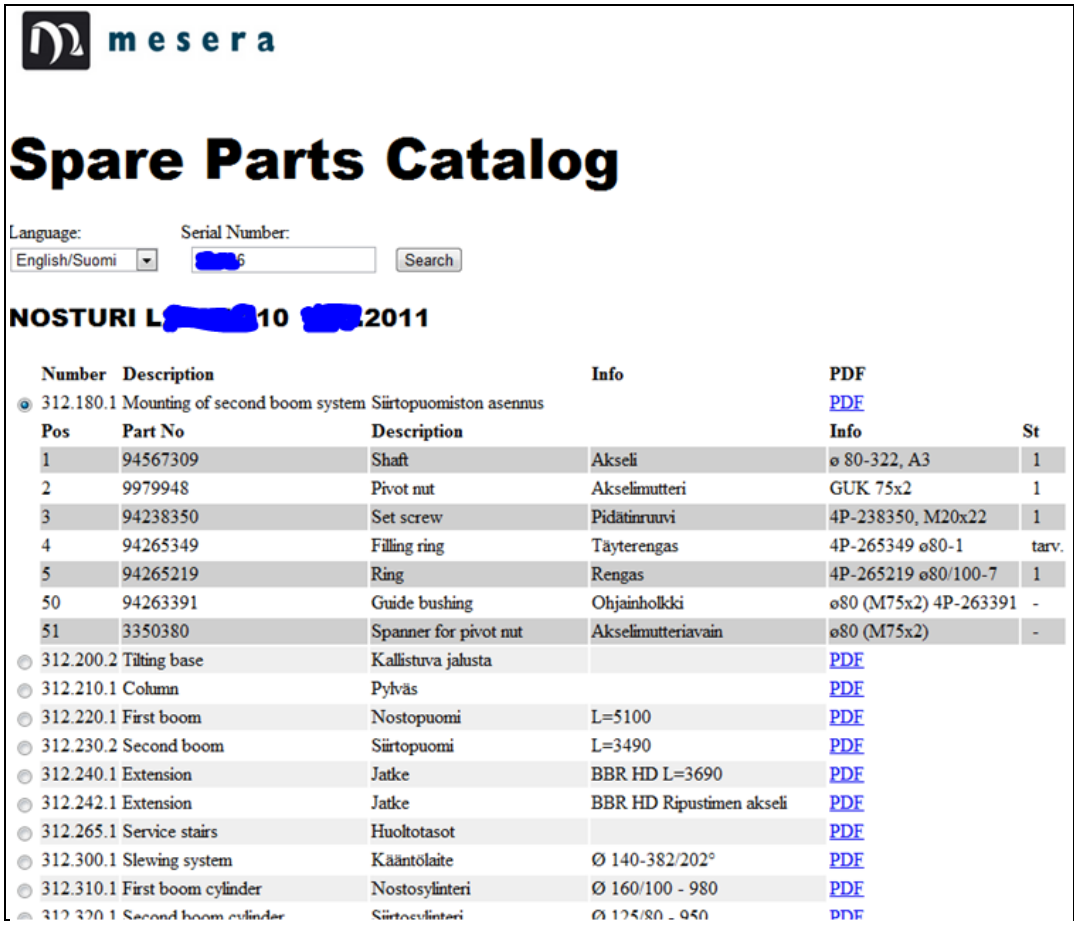
Kuvassa 10 on esitetty tilanne, kun haetulla työnumerolla ei löydy tuotetta tietokannasta.



Kuva 10. Virheilmoitus. Tuotetta ei löydy.

Uusi tapa hallinnoida sallittuja ohjelman käyttäjiä helpotti käyttäjälistan ylläpitämistä. Nyt ohjelmalla oli oma Windows-käyttäjäryhmä, jolla oli kaikki tarvittavat oikeudet. Piti vain lisätä tai poista käyttäjä ryhmästä, jotta oikeutetut käyttäjät olisivat aina ajan tasalla. Lisäsin oman käyttäjätunnuksen ja testikäyttäjän ryhmään vuorotellen ja testasin toimivuutta.

Kuvassa 11 on ohjelman lopullinen käyttöliittymäversio. Se esittää sitä ohjelman toimivuusastetta, joka on nimetty ohjelman versioksi 1.0.



mesera

Spare Parts Catalog

Language: English/Suomi

Serial Number: [redacted] Search

NOSTURI L[redacted]10 [redacted]2011

Number	Description		Info	PDF	
312.180.1	Mounting of second boom system	Siirtopuomiston asennus		PDF	
Pos	**Part No**	**Description**		**Info**	**St**
1	94567309	Shaft	Akseli	ø 80-322, A3	1
2	9979948	Pivot nut	Akselimutteri	GUK 75x2	1
3	94238350	Set screw	Pidätinruuvi	4P-238350, M20x22	1
4	94265349	Filling ring	Täyterengas	4P-265349 ø80-1	tarv.
5	94265219	Ring	Rengas	4P-265219 ø80/100-7	1
50	94263391	Guide bushing	Ohjainholkki	ø80 (M75x2) 4P-263391	-
51	3350380	Spanner for pivot nut	Akselimutteriavain	ø80 (M75x2)	-
312.200.2	Tilting base	Kallistuva jalusta		PDF	
312.210.1	Column	Pylväs		PDF	
312.220.1	First boom	Nostopuomi	L=5100	PDF	
312.230.2	Second boom	Siirtopuomi	L=3490	PDF	
312.240.1	Extension	Jatke	BBR HD L=3690	PDF	
312.242.1	Extension	Jatke	BBR HD Ripustimen akseli	PDF	
312.265.1	Service stairs	Huoltotasot		PDF	
312.300.1	Slewing system	Kääntölaite	Ø 140-382/202°	PDF	
312.310.1	First boom cylinder	Nostosylinteri	Ø 160/100 - 980	PDF	
312.320.1	Second boom cylinder	Siirtosylinteri	Ø 125/80 - 950	PDF	

Kuva 11. Lopullinen versio ohjelmasta

Toimeksiantaja on hyväksynyt ohjelman ja päätettiin, että tässä vaiheessa ei ruveta enää lisäämään ominaisuuksia ohjelmaan, vaan aloitetaan ohjelman oikea käyttö. Seuraava askel oli järjestää ohjelman datan ajantasaisuus ja muut käyttöönottoon liittyvät toimenpiteet.

6 KÄYTTÖÖNOTTO

6.1 Käyttöönoton suunnittelu

Koska ohjelma oli hyväksytty tuotantokäyttöön, käyttöönotto piti suunnitella todella tarkasti. Testiympäristössä käytettiin dataa, joka oli manuaalisesti otettu kopio toisen yrityksen palvelimella sijaitsevasta oikeasta tietokantadatasta. Koska ohjelman lopulliseksi toimintaympäristöksi oli valittu nimenomaan Saides Oy:n palvelin, piti järjestää datan automatisoitu päivittyminen. Sen lisäksi tässä vaiheessa ohjelman luotettavuuteen ja turvallisuuteen piti kiinnittää erityistä huomiota. Kaikki mahdolliset virhetilanteet oli huomioitava ja piti tehdä niin, että missään nimessä virhetilanteen syntyessä käyttäjälle ei paljastuisi sellaisia tietoja, joiden pohjalta hän saisi tietoja ohjelman koodista tai tietokantayhteydestä. Myös käyttöönoton viimeinen vaihe eli oikeiden käyttäjien lisääminen ohjelmaan oli suunniteltava niin, että ne olisivat helposti hallinnoitavissa ja eriteltävissä muista Saides Oy:n domain-käyttäjistä.

Hahmoteltuani kaikki käyttöönottoon liittyvät toiminnot samalla hahmottui niiden toteutusjärjestys. Ensimmäiseksi suunnittelin varsinaiseen ohjelman liittyvät turvallisuusjärjestelyt mahdollisten virhetilanteiden varalta. Sen jälkeen oli vuorossaan tietokantadatan päivitys. Päätin lisätä vasta sen jälkeen oikeat käyttäjätunnukset. Näiden toimintojen jälkeen voitaisiin pitää ohjelmaa täydellisesti käyttöönotettuna ja projektia valmiina.

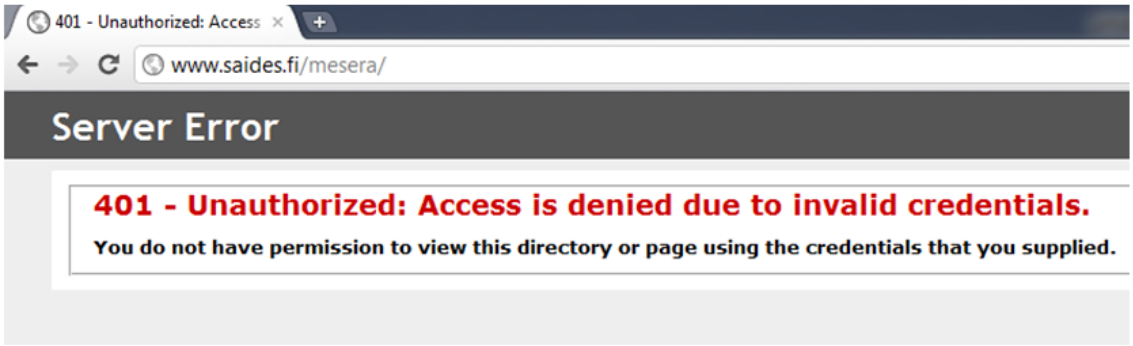
6.2 Ohjelman turvallisuuden parantaminen

Ensimmäiseksi siirsin kaikki ohjelmaa varten tehdyt tietokantanäkymät suoraan ohjelman koodiin. Tein tämän sen takia, että näin ohjelman hallinnointi helpottuisi ja tietokannan päivitykset ja muutokset eivät vaikuttaisi niin paljon ohjelman toimivuuteen. Muutoksen takia jouduin paljon muokkaamaan kyselyitä ohjelmassa, koska nyt saman kyselyn sisään piti tuoda taulusta ”Vocabulary” sekä

englanninkielinen versio nimityksestä, että valitun kielen mukainen. Tämä onnistui sillä, että otin alias käyttöön.SELECT-lauseeseen tulivat kentät vocEN.text, voc.text, jotka molemmat viittaavat samaan ”Vocabulary”-tauluun, mutta WHERE-ehdossa toinen niistä ottaa arvon valitusta kielestä ja toinen on muotoa vocEN.Lang=’EN’.

Sen jälkeen testasin uudemman kerran, miten kaikki ohjelman sisäiset virheilmoitukset Try-Catch Error -rakenteissa toimivat ja millaiset tulokset käyttäjä näkee, jos vaikka tietokantayhteys ei toimi tai se on virheellinen. Ohjelma käyttäytyi juuri niin, kuin oli suunniteltu ja antoi käyttäjälle hakutulosten ja IIS-virhesivun sijasta lyhyet virheilmoitukset tyyliin ”DB error 2. Can not connect to database.”

Seuraava askel oli muuttaa sivun IIS-asetuksissa virheilmoitukset niin, että käyttäjälle näkyisivät vain lyhyet virheilmoitukset ilman tarkkoja selityksiä, joista voisi saada selväksi sivun kansiorakenteen ym. tiedot, joita ei ole tarkoitettu käyttäjien tietoon. Kuvassa 12 on esimerkki siitä, miltä käyttäjälle näkyvät IIS-virheilmoitukset näyttävät.



Kuva 12. IIS-virheilmoitus

6.3 Tietokantojen päivityksen järjestäminen

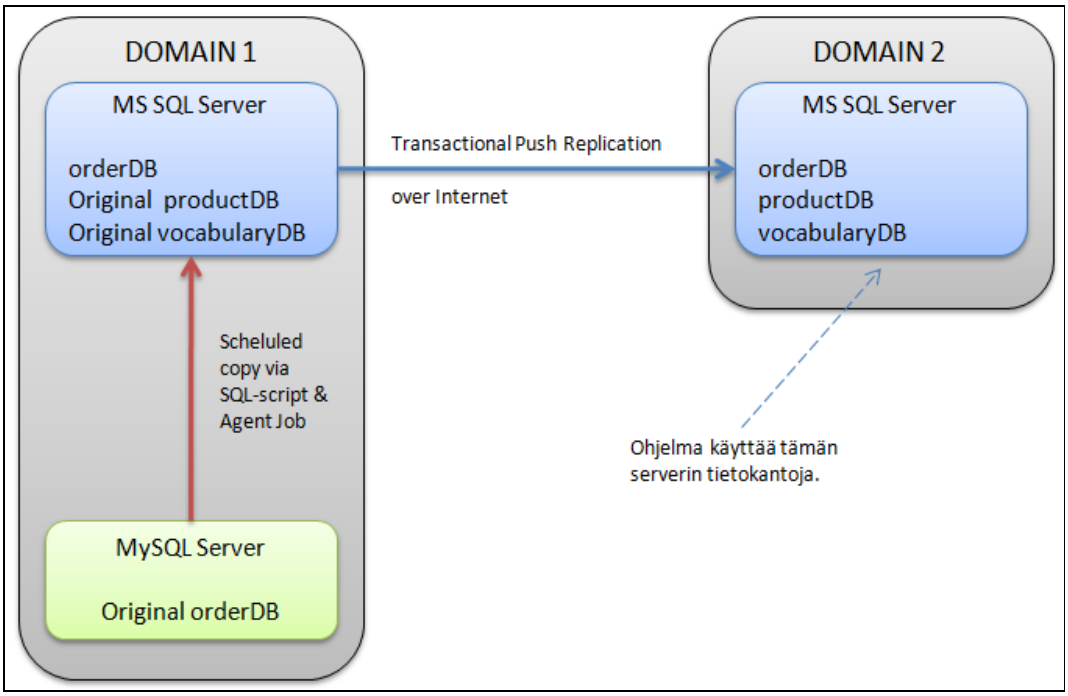
Data oli jaettu sillä tavalla, että alkuperäiset tietokannat ”productDB” ja ”vocabularyDB” olivat MS SQL-serverillä ja ”orderDB”-tietokanta MySQL-serverillä sa-

man domainin sisällä. Yhteys doimainiin järjestyi VPN-yhteydellä. Päätimme yhdessä projektin vetäjän kanssa, että ensin minä järjestän datan kopioitumisen ensin MySQL-serveriltä saman domainin MS SQL-serverille ja sitten kaikkien tietokantojen replikointi Saides Oy:n serverille.

Tietokannan ”orderDB” datan kopioituminen onnistui helposti, sillä MS SQL-serverillä oli jo valmiiksi MySQL-serveri linkitettynä. Tein SQL-skriptin SQL Server Agenttiin, joka tarkisti ensin yhteyden tietokannan tauluihin ”SELECT COUNT(*) FROM…” -lauseen avulla ja jos kysely meni läpi ongelmitta, poisti vanhat rivit kopiotietokannasta ja lisäsi kaikki rivit tauluihin oikeasta MySQL-tietokannasta. Ajastin skriptin ajettavaksi kerran päivässä. Koska koko skriptin ajoaika jäi alle kahden minuutin, ja toimi moitteettomasti, pidin ratkaisua hyvänä.

Replikoinnista minulla ei ollut kokemusta, joten päätin ottaa selvää alan kirjoista. Hyviä ohjeita löydettyäni päätin kokeilla järjestää Transactional-replikoinnin (Stanek 2010, 483). Näin data päivittyisi lennossa ja olisi aina ajan tasalla. Ensin kokeilin Pull Subscription -vaihtoehtoa (Stanek 2010, 512). Sain helposti ohjeiden mukaisesti toimimaan Publisher ja Distributor, mutta Subscriber ei jostain syystä saanutkaan toimimaan. Siinä vaiheessa, kun sen piti lähteä kopioimaan rivejä, tuli virheilmoitus ”OS error 1326”. Selailin lukuisia foorumeja, mutta en löytänyt ratkaisua siihen.

Toisena vaihtoehtona olisi järjestää Push Subscription, mutta siihen tarvittaisiin yhteys Publisher-serveriltä Subscriber-serverille, jota ei VPN-yhteys sallinut. Koska Saides Oy:n serveriin pääsi tekemään yhteyden myös Internet-verkon kautta, päätimme kokeilla sitä vaihtoehtoa. Sitä varten tilattiin palomuurin tarjoajalta tarvittavien porttien avaus. Kun saimme portit auki, yhteys Publisher-serveriltä Subscriber-serverille alkoi toimia. Sen jälkeen järjestin kaikki tarvittavat replikoinnit Transactional Push Subscription -toiminnon kautta. Näin kaikki alkuperäisellä serverillä tapahtuvat muutokset kopioituvat Saides Oy:n serverille saman tien ja näin data on aina ajan tasalla. Kuva 13 havainnollistaa, millä tavalla olen järjestänyt datan ajantasaisuuden Saides Oy:n serverillä.



Kuva 13. Tietokantojen päivittäminen Saides Oy:n palvelimelle

6.4 Käyttäjien lisääminen

Oikeiden käyttäjien lisääminen piti järjestää niin, että käyttäjälista olisi jälkeenpäin helposti muokattavissa ja ylläpidettävissä. Tein Saides Oy:n palvelimelle uuden virtuaalisen organisaatioyksikön (Active Directory Organisational Unit), johon uudet käyttäjät olisi lisättävä. Ohjelman käyttäjäryhmä oikeuksineen oli jo valmiiksi olemassa. Uuden käyttäjän lisäämisen jälkeen piti vain lisätä se ryhmään ja näin hän sai oikeudet ohjelman käyttöön.

Esivalmisteluiden jälkeen Saides Engineering Oy:n toimitusjohtaja itse lisäsi käyttäjät domainiin ja ryhmään ja antoi niille salasanat. Käyttäjän kommenttikenttään oli laitettu sähköpostiosoite, johon uusi salasana lähetetään. Tässä vaiheessa käyttäjätunnuksia ei ollut paljon ja toimitusjohtaja lupasi manuaalisesti hoitaa salasanojen vaihdon ja jakelun. Koska sähköpostiosoitteet olivat

työpaikkakohtaisia, tätä jakelutapaa voidaan pitää turvallisena, sillä yleensä, kun henkilöllä loppuu työsopimus yritykseen, sen sähköpostilaatikko myös poistetaan.

7 ARVIOINTI JA OHJELMAN JATKOKEHITYKSEN IDEOINTI

7.1 Projektin arviointia

Projektin tavoitteena oli saada toimiva varaosaluettelo Saides Oy:n tietokantojen pohjalle. Projekti eteni sulavasti määrittelystä käyttöönottoon. Feature Driven Development -kehitysmalli osoittautui hyväksi ratkaisuksi tilanteessa, kun asiakkaalla oli vain visio siitä, miltä lopputuloksen pitää näyttää. Jatkuva vuorovaikutus projektin vetäjän ja asiakkaan kanssa auttoi askel askeleelta edetä suunnitteluvaiheesta käyttökelpoiseen ohjelmaan.

Toimintolistaan (taulukko 1) määritellyist toiminnoista on toteutettu noin puolet, joten ei voida sanoa, että projektin aikana olen toteuttanut kaikki asiakkaan toiveet. Toisaalta saatiin toimiva ohjelma, joka on otettu tuotantokäyttöön. Sain myös kuulla, että en ollut ensimmäinen henkilö, joka yritti tehdä siihen varaosaluetteloon tarkoitettua ohjelmaa. Ennen minua on jo kaksi kertaa yritetty samaa projektia ja erilaisten syiden takia sitä ei oltu saatu silloin valmiiksi.

Sen takia, että ohjelmani on hyväksytty käyttöön ja sain positiivista palautetta, uskallan sanoa, että projektini on onnistunut hyvin. Olen myös mielestäni toteuttanut varmatoimisen, turvallisen ja luotettavan ohjelman. Turvallisuutta ja tietojen luotettavuutta on tarkistettu useaan otteeseen, sekä minä itse olen testannut, että toimeksiantajan testikäyttäjät. Ohjelman ylläpito ja hallinnointi on myös järjestetty hyvin. Tämä tuli jo esille silloin, kun Saides Engineering Oy:n toimitusjohtaja itse lisäsi uudet ohjelman käyttäjät.

Projektin aikana olen oppinut todella paljon uutta. Microsoft-ympäristössä ohjelmointi ja asiakassuuntainen ajattelu ovat mielestäni tärkeimpiä oppimia uusia taitoja. Ennen tätä projektia minulla ei ollut kokemusta VB.NET-ohjelmoinnista, mutta olen hyvin nopeasti oppinut sen. Tässä projektissa opin myös, että työn tuloksen (ohjelman, nettisivun, tietokannan ym.) loppukäyttäjän tyytyväisyys on

aina ensisijalla. Loppukäyttäjän kannalta ei ole olennaista, mitä menetelmiä on käytetty, vaan miten hän kokee käyttäjäkokemuksen.

7.2 Ohjelman jatkokehityksen ideoita

Saides Engineering Oy:n toimitusjohtajalla sekä Meseran tiimillä oli paljon hyviä ideoita ohjelman toiminnallisuuteen suhteen, joita ei ole ehditty toteuttaa. Suurin osa niistä näkyy jo projektin määrittelyn toimintolista-osiossa. Jos minulla olisi mahdollisuus jatkaa projektin kehittämistä, asettaisin jatkokehitysvaiheet seuraavaan järjestykseen.

Ensimmäisenä yrittäisin tehdä ohjelman, joka automaattisesti luo uudet käyttäjien salasanat ja jakelee ne sähköpostiosoitteisiin säännöllisin väliajoin. Jos lähettäminen ei onnistu sen takia, että sähköpostiosoitetta ei enää ole olemassa, ohjelma poistaisi käyttäjätunnuksen. Näin saataisiin automatisoitua käyttäjälistan ylläpitoa.

Seuraavana tekisin ohjelmaan haun tuotenimen mukaan. Se vaatisi myös hakutulosten ryhmittelyä, koska hakutulosrivejä saattaisi olla yli 100 kappaletta. Sen lisäksi voisi tehdä luettelon kaikista tuotteista, josta valitsemalla pääsee katsomaan valitun tuotteen moduulit, osat ja muut tiedot. Tämä muutos parantaisi ohjelman käytettävyyttä olennaisesti.

PDF-dokumentin luominen online-tilassa suoraan tietokannasta olisi seuraava askel. Luultavammin tämä vaatisi uuden ohjelman luomista. Tämä helpottaisi paljon Saides Oy:n tekijöiden työtä, koska ei tarvitsisi pitää huolta siitä, että uudesta tietokannassa olevasta tuotteesta löytyy myös valmiit PDF-versiot joka kieltä kohden.

Kännykkäversion ja verkkokaupan tekeminen vaatisivat jokainen omaa isoa projektia. Nykykännykät eivät tue Windows-käyttäjätunnuksiin perustuvaa kirjautumista. Tämän takia pitäisi ehkä luoda suurimpia kännykkäkäyttöjärjestelmiä varten (Android, iOS, Windows, Symbian) omat ohjelmat. Toinen vaihtoehto

olisi ottaa käyttöön toinen kirjautumistapa, joka vaatisi kirjautumistapojen turvallisuuden ja hallinnoinnin tutkimista.

Verkkokauppa varaosaluettelosta olisi laajuudeltaan iso projekti. Mukaan pitäisi huomioida mm. tietokantaan tulevat muutokset, verkkokaupan sivuston turvallisuus ja käyttäjätietojen salaus, verkkomaksutapoihin liittyvät toiminnot ja riskit ja käytettävään myyntijärjestelmään liittäminen.

LÄHTEET

Agile Manifesto 2001. Principles behind the Agile Manifesto. Viitattu 07.11.2011 http://agilemanifesto.org/principles.html

Amber S. W. 2003 Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development. Viitattu 07.11.2011 http://www.agilemodeling.com/essays/amdd.htm

De Luca J. 2003. Issue 2 – How to Reduce the Risk of Fixed-Price Estimates. Viitattu 07.11.2011 http://www.featuredrivendevelopment.com/node/527

Khramtchenko S. 2004. Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments. Cambridge, Massachusetts, USA: Harvard University

MacDonald, M. 2010. Beginning ASP.NET in VB 2010, New York, USA: Apress.

MacDonald, M.; Madbutt, D. & Freeman, A. 2010. Pro ASP.NET 4 in VB 2010, New York, USA: Apress.

Nebulon 2005. Feature Driven Development (FDD) Overview Presentation. Viitattu 07.11.2011 http://www.nebulon.com/articles/fdd/download/fddoverview.pdf

Pressman R. S. 1994. Software engineering. London, UK: McGraw-Hill Book Company.

Sommerville I. 2007. Software engineering 8. Harlow, England: Pearson Education Limited

Spaanjaars, I. 2010. Beginning ASP.NE T 4: in C# and VB. Indianapolis, Canada: Wiley Publishing, Inc.

Stanek William R. 2010. Microsoft SQL Server 2008 Administrator Pocket Consultant (2nd). Redmond, Washington, USA: Microsoft Press.

Tian J. 2005. Software Quality Engineering Testing, Quality Assurance, and Quantifiable Improvement. New Jersey, USA: John Wiley & Sons, Inc.