TAMPEREEN
AMMATTIKORKEAKOULU

LIIKETALOUS

TUTKINTOTYÖRAPORTTI

XML -TOIMINTOJEN KEHITTÄMINEN SOLTEQ MERXISSÄ

Niilo Ruotsalainen

Tietojenkäsittelyn koulutusohjelma
Huhtikuu 2005
Työn ohjaaja: Jyrki Vehmas

TAMPERE 2005

TAMPEREEN
AMMATTIKORKEAKOULU

LIIKETALOUS

Tekijä(t): Niilo Ruotsalainen

Koulutusohjelma(t): Tietojenkäsittely

Tutkintotyön nimi: XML -toimintojen kehittäminen Solteq Merxissä

Title in English: Developing XML Functions of Solteq Merx

Työn valmistumis-kuukausi ja -vuosi: Huhtikuu 2005

Työn ohjaaja: Jyrki Vehmas **Sivumäärä:** 47

TIIVISTELMÄ

Yritykset pyrkivät tekemään yhä tiiviimpää yhteistyötä tehostaakseen toimintojaan. Sähköinen liiketoiminta ja tietojen vaihto organisaatioiden välillä ovat tärkeässä asemassa tässä kehityksessä. Siirrettävien tietojen määrä ja muoto asettaa yhä tiukempia vaatimuksia tiedonsiirroille. XML -teknologia tarjoaa joustavan ja kilpailukykyisen tavan näiden modernien tiedonsiirtojen rakentamiseen.

Tutkintotyön toimeksiantaja on ohjelmistoja ja muita IT-palveluita tuottava yritys Solteq Oyj. Solteq Merx on toimeksiantajan kehittämä toiminnanohjausjärjestelmä tukkukaupan ja vähittäiskaupan toimialoille. Järjestelmä toimii IBM iSeries suurtietokone laitealustalla.

Tavoitteena tässä työssä on kehittää Solteq Merxiin valmius vastaanottaa ja muodostaa XML -muotoisia tietoja. Tähän tavoitteeseen pyritään työssä esiteltävällä sovelluskokonaisuudella.

Solteq XML rajapinnaksi nimetty sovelluskokonaisuus on työkalu XML -tiedonsiirtojen rakentamiseen. Sen toiminnallisuuteen kuuluu XML -sanoman muototarkastuksen tekeminen ja tietojen purkaminen DB2 relaatiotietokantaan sekä XML -sanoman muodostaminen tietokannasta luetuista tiedoista. Sovelluskokonaisuus toimii kaikissa moderneissa iSeries ympäristöissä.

Sovelluskokonaisuuden esittelyn lisäksi työssä tutkitaan XML:n ominaisuuksia tiedonsiirtomuotona, jotta uusia ominaisuuksia osattaisiin hyödyntää tulevissa tiedonsiirtohankkeissa. Tutkimusosiota varten on työssä kerätty tietoja eri lähteistä sekä suoritettu vertailua XML:n ja Solteq Merxin perinteisten tiedonsiirtomuotojen välillä.

Tutkimuksen perusteella voidaan todeta, että XML tiedonsiirtomuotona soveltuu parhaiten kohteisiin, joissa siirrettävä tietosisältö tarvitsee rakenteen kuvausta, siirrettävän tiedon määrä ei ole todella suuri ja siirron osapuolena olevissa järjestelmissä tai toisessa niistä on selainkäyttöliittymä. Tiedonsiirtomuotojen vertailussa ilmeni, että suuria tietomääriä sisältävät tiedonsiirrot kannattaa toteuttaa esim. EDIFACT -standardiin perustuvalla tekniikalla. Valinta XML:n ja muiden tiedonsiirtomuotojen välillä perustuu kuitenkin lähes aina kustannuksiin eikä soveltuvuuteen.

Tutkintotyössä aloitettua kehitystyötä jatketaan toimeksiantajaorganisaatiossa tulevaisuudessakin. Kehitys pohjautuu tässä työssä esiteltyihin tutkimustuloksiin ja toteutettuun ratkaisuun. XML -teknologian kehittymistä täytyy kuitenkin seurata jatkossakin. Erityisesti tulee seurata mitkä XML:n standardit tulevat vakiintumaan käytössä ja rakentaa valmiit tiedonsiirtokomponentit näitä varten.

Avainsanat: XML,EDI,ohjelmointi

Sisällysluettelo

1 JOHDANTO **5**

2 TAUSTAA TUTKINTOTYÖLLE **6**

2.01 TAVOITTEET JA TOIMEKSIANTAJA 6
2.02 TUTKINTOTYÖN OSA-ALUEET 7
2.03 SOLTEQ OYJ 7
2.04 SOLTEQ MERX -TOIMINNANOHJAUSJÄRJESTELMÄ 8
2.05 SOLTEQ JA XML 9

3 TÄRKEITÄ KÄSITTEITÄ **9**

3.01 IBM ISERIES -LAITEALUSTA 9
3.02 OS/400 KÄYTTÖJÄRJESTELMÄ 9
3.03 INTEGRATED LANGUAGE ENVIRONMENT 9
3.04 ILE RPG -OHJELMOINTIKIELI 10
3.05 XML 11
3.06 XML -PARSERI 14
3.07 XML TOOLKIT FOR ISERIES 15
3.08 EDI/OVT 15

4 EDI/OVT LIITTYMÄT SOLTEQ MERXISSÄ **17**

5 XML – TIEDONSIIRTOMUOTONA **18**

5.01 OMINAISUUDET 19
5.01.1 Tiedon ymmärrettävyys *19*
5.01.2 Tiedon rakenne *20*
5.01.3 Tiedon siirtäminen *20*
5.01.4 Tiedon käsittely *20*
5.02 LIITTYMÄN RAKENTAMINEN 21
5.02.1 Järjestelmävaatimukset *22*
5.02.2 Liittymän suunnittelu *22*
5.02.3 Liittymän toteutus *23*
5.02.4 Yhteenveto XML -tiedonsiirroista *25*

6 XML JA EDI -STANDARDIT **25**

6.01 XML VS. EDI STANDARDIT 26
6.01.1 Sähköisen kaupankäynnin standardit *26*
6.01.2 Tiedon esitystavat ja käytettävyys *26*
6.01.3 Tiedon käsittely *27*
6.01.4 Yhteenveto XML:n ja EDI -standardien vertailusta *28*
6.02 XML/EDI TULEVAISUUDEN RATKAISU? 29

7 XML -TOIMINNOT SOLTEQ MERXISSÄ **30**

7.01 LÄHTÖTILANTEEN SELVITYS 31
7.02 XML - TOIMINTOJEN KEHITYS 31
7.03 XML – TOIMINTOJEN TULEVAISUUS 33

8 XML -RAJAPINTA SOLTEQ MERXIIN **33**

8.01 VAATIMUSMÄÄRITTELY 34
8.01.1 XML -tiedoston muodostaminen *34*
8.01.2 XML -tiedoston purkaminen *35*
8.01.3 Kokonaisuuden ylläpidettävyys *35*
8.01.4 Uusien XML -liittymien rakentaminen rajapintasovelluksen avulla *36*
8.02 TEKNINEN SUUNNITTELU 36
8.02.1 Kokonaisuuden jakaminen toiminnallisiin moduuleihin *36*
8.02.2 XML Toolkitin toimintojen käyttäminen *36*
8.02.3 Tietojen lukeminen ja kirjoittaminen tietokantaan *37*

8.03 TOTEUTUKSEN KUVAUS 38
8.03.1 XML -dokumentin muodostus 38
8.03.2 XML -dokumentin purku 39
8.04 XML -LIITTYMÄN RAKENTAMINEN TOTEUTUKSEN AVULLA 41
8.04.1 XML -sanoman lähetys 41
8.04.2 XML -sanoman vastaanotto 42

9 LOPPUTULOS JA ARVIOINTI 44

9.01 LOPPUTULOS 44
9.02 LOPPUTULOKSEN ARVIOINTI 44

LÄHTEET 46

1 Johdanto

Nykyaikaisen tietojenkäsittelyn tärkeimpiä ja kasvavimpia toimintoja ovat tietojen automaattiset siirrot järjestelmästä toiseen. Perinteisesti tämä tarkoittaa lähettävässä järjestelmässä muodostettua konekielistä sanomaa, joka siirretään vastaanottavaan järjestelmään, jossa sanoma puretaan järjestelmän tietokantaan. Tänä päivänä XML -teknologia tarjoaa tähän joustavamman ja edullisemman vaihtoehdon.

XML -tiedonsiirtoon perustuvien järjestelmien välisien liittymien määrä ja kysyntä on kokoajan kasvussa. Siksi toiminnanohjausjärjestelmiä toimittavien yritysten on perehdyttävä tähän teknologiaan ja pyrittävä tarjoamaan omat ratkaisunsa näihin toimintoihin.

Tutkintotyöni käsittelee Solteq Merxin XML -toimintojen kehittämistä. Solteq Merx on IBM iSeries suurtietokoneympäristöön kehitetty tukku- ja vähittäiskaupan toimialoille suunnattu toiminnanohjausjärjestelmä. iSeries -ympäristö antaa työlle omat haasteensa, koska perinteisten tiedonsiirtoihin liittyvien ongelmien lisäksi joudun työssäni pohtimaan ympäristön tuomia lisähaasteita. Kehitystyön alkutilanteessa olemassa olevat XML -toiminnot olivat kokeiluasteella ja Merxin parissa toimivassa organisaatiossa ei ollut XML -teknologian syvällistä tuntemusta.

Uuden teknologian käyttöönotto sovelluskehityksessä vaatii aina asioiden tutkimista ja teorioiden testaamista käytännössä. Lopputuloksen kannalta aika on merkittävä asia. Mitä paremmin toteutus suunnitellaan, sitä parempi on yleensä myös lopputulos. Tässä tutkintotyössä alkututkimus tarkoittaa XML:n ominaisuuksien selvitystä ja niiden vertaamista järjestelmässä olemassa olevissa liittymissä käytettyihin tiedon esitysmuotoihin. Selvityksellä pyritään luomaan käsitys XML:n mahdollisista käyttökohteista ja saatavista hyödyistä.

XML tiedonsiirron rakentaminen iSeries -ympäristössä tarkoittaa tietojen keräämistä tietokannasta ja kirjoittamista XML -muotoiseen tiedostoon tai päinvastoin. Perusongelma tiedonsiirroissa iSeries ympäristön ja esimerkiksi PC -järjestelmien välillä ovat merkistöeroavaisuudet. XML:ssä tätä ongelmaa ei esiinny, koska tiedot ovat siinä kaikkien järjestelmien ymmärtämässä muodossa. Toisaalta XML -tiedosto on tietovirtatiedosto ja Merxissä käytetyissä ohjelmointikielissä ei ole tämän tietotyypin tukea. Tämä ongelma on tässä työssä ratkaistu käyttämällä soveltuvaa XML -ohjelmointirajapintaa.

Työn lopputuloksena esitellään sovelluskokonaisuus, jonka avulla XML -tiedonsiirtojen toiminnallisuudet toteutetaan käytössä olevilla työkaluilla. Lisäksi käydään läpi käytetyt ohjelmointitavat ja kerrotaan kuinka näistä komponenteista muodostetaan toimiva tiedonsiirto.

2 Taustaa tutkintotyölle

Tiedonsiirrot

Yritykset pyrkivät tehostamaan toimintojaan ja säästämään kustannuksia tekemällä yhteistyötä. Yritysten välisen verkottumisen edellytys on tietojen vaihto. Tänä päivänä tietoverkkojen siirtonopeuksien kasvu mahdollistaa suurten tietomäärien ja monimuotoisten sisältöjen siirtämisen sähköisesti organisaatioiden välillä. Sähköisen viestinnän määrän ja laadun kasvaessa vaaditaan yritysten tietojärjestelmiltä kykyä vastaanottaa ja lähettää monenlaisia ja muotoisia tietoja. (Arto Mertaniemi 1999.)

Tämä vuonna 1999 kirjoitettu teksti pitää edelleen paikkaansa. XML on tullut osaksi nykyaikaisia tietojärjestelmiä ja niiden välisiä tiedonsiirtoja. XML sopii monikäyttöisenä tietomuotona hyvin nykyaikaisten tiedonsiirtojen vaatimuksiin. XML:n käyttö on koko ajan lisääntynyt ja tulevaisuudessa sen merkitys tiedon liikkumissa ja esittämisessä tulee edelleen kasvamaan. Tämän vuoksi yritysten operatiivissa liiketoiminnoissa toimivien tietojärjestelmien on pystyttävä lukemaan ja kirjoittamaan XML -muotoista tietoa.

2.01 Tavoitteet ja toimeksiantaja

Toimeksianto

Tutkintotyön tavoite syntyi toimeksiantaja Solteq Oyj:n tarpeesta saada Solteq Merx toiminnanohjausjärjestelmään valmius lähettää ja vastaanottaa XML -muotoista tietoa. Tämä ei kuitenkaan ole vielä Solteqin kannalta riittävä tavoite, sillä uusia toimintoja pitää pystyä hyödyntämään jatkossa useissa asiakasprojekteissa ja myös markkinoimaan asiakkaille. Tätä varten on tutkittava XML:n ominaisuuksia tiedonsiirtomuotona ja selvitettävä sen perusteella minkä tyyppisissä tiedonsiirroissa Solteq Merxissä XML:n käytöstä saadaan etuja, jotka parantavat asiakkaan sähköistä liiketoimintaa. Pitemmällä aikavälillä Solteq haluaa XML toimintojen lisäävän järjestelmän arvoa.

Tavoitteet

Saavuttaakseen toimeksiantajan tavoitteen on työssä rakennettava sovellus tai sovelluskokonaisuus, jossa on huomioitu uudelleen käytettävyys Solteq Merxin ja muiden järjestelmien välisissä tiedonsiirroissa. Toteutuksen vaatimuksena on siis monikäyttöisyys, eli sen on pystyttävä käsittelemään useimmat tiedonsiirroissa käytetyt XML -tiedostomuodot. Ratkaisussa on myös otettava huomioon Solteq Merxin laitteiston ja käyttöjärjestelmäympäristön erikoisvaatimukset.

Jotta edellä mainitusta toteutuksesta saadaan paras hyöty irti, on työssä verrattava XML:n ominaisuuksia tiedonsiirtomuotona järjestelmässä pääasiassa käytettyjen tiedonsiirtomuotojen ominaisuuksiin. Vertailun tarkoituksena ei ole pelkästään XML:n etujen kartoitus, vaan yhtälailla huomioida sen heikkoudet. Tältä osin tutkintotyön tavoitteena on selvittää missä XML:n käytöstä saadaan selkeää hyötyä ja missä sitä ylipäätään voidaan käyttää. Tämä selvitys tehdään omana osionaan tutkin-

totyöhön ja tästä halutaan saada perusteita myös toteutusosan suunnitteluun.

2.02 Tutkintotyön osa-alueet

Tämä tutkintotyö on jaettu kahteen erilaiseen osa-alueeseen, joilla on kummallakin omat tavoitteensa. Ensimmäinen osa-alue tarjoaa tutkimustuloksen XML:stä tiedonsiirtomuotona ja toinen osa esittelee XML:ää käsittelevän sovelluskokonaisuuden. Osa-alueet liittyvät toisiinsa siten, että tutkimus osan tuloksia on käytetty hyväksi toteutusosan suunnittelussa.

Tutkimusosa

Tutkintotyön ensimmäinen osa-alue käsittää XML:n ominaisuuksien selvityksen ja vertailun perinteisiin tiedonsiirtomuotoihin. Samalla selvitetään mitä erikoisvaatimuksia IBM iSeries ympäristö aiheuttaa XML:n käsittelylle. Tulokset tästä esitetään kirjallisesti tässä raportissa.

Toteutusosa

Toteutusosa esittelee tutkintotyön tavoitteissa määritellyn XML:ää käsittelevän sovelluskokonaisuuden. Tässä osassa käydään läpi toteutuksen vaatimusmäärittely, toteutuksen suunnittelu ja esitellään ratkaisu.

2.03 Solteq Oyj

Toimeksiantaja

Solteq Oyj on Tamperelainen ohjelmistotalo, joka työllistää n. 230 henkilöä. Solteq Oyj tuottaa strategisia, eri liiketoiminta aloille suunnattuja tietojärjestelmiä. Yrityksen päätoimialueet ovat kauppa, autokauppa ja tiettyjen teollisuussegmenttien toiminnanohjausjärjestelmät. Toimialaratkaisujen lisäksi Solteqilla on asiakaskohtaisia yhteistyösopimuksia joiden perusteella kehitetään asiakkaan tietojärjestelmää jatkuvana prosessina.

Solteq Oyj aloitti toimintansa nimellä TH Tiedonhallinta Oyj vuonna 1982. Helsingin arvopaperipörssiin yhtiö listautui vuonna 1999. Solteq Oyj:n kotipaikka on Tampere. Muut toimipisteet ovat Helsingissä, Lahdessa, Hämeenlinnassa ja Kuopiossa.

Tuotteet ja palvelut

Solteq tarjoaa asiakkailleen ohjelmistoja, palveluja ja laitteita. Omaa ratkaisutarjontaa täydennetään yhteistyökumppaneiden ohjelmistoilla ja teknologialla. Tärkeimmät kumppanit ovat alojensa markkinajohtajat SAP ja IBM. ”Solteqin, SAP:n ja IBM:n yhteistyö tarjoaa kotimaisille keskisuurille ja kasvaville yrityksille ratkaisukokonaisuuden, jossa hyödynnetään SAP:n ja IBM:n kansainvälistä järjestelmä- ja teknologiaosaamista sekä Solteqin toimialaosaamista ja asiantuntemusta paikallisilla markkinoilla.” (Solteq Oyj:n Internet sivut 2004.)

Toimialat

Solteqin asiakaskunta on toimintavuosien saatossa kasvanut teknisen tukkukaupan ja autokaupan aloilta myös metsäteollisuuteen, asiakaskohtaisiin ratkaisuihin ja vähittäiskauppaan. Muutamia Solteqin asia-

kasyrityksiä vapaasti poimittuna: UPM Kymmene/Schauman Wood Oy, Berner Oy, Notex Yhtiöt Oy (Vapaavalinta), Oy Starkki Ab, Nissan, Veho Group Oy Ab, Töölön Matkatoimisto Oy.
(Solteqin vuosikertomus 2004.)

Solteqin tavoitteena on olla mukana kehittämässä asiakkaan liiketoimintaa toimittamalla kokonaisvaltaisesti kaikkia IT -palveluita, toimialan tai asiakaskohtaisen tarpeen mukaan. Pyrkimyksenä on luoda pitkäaikaisia ja kumppanuutta korostavia asiakassuhteita.

Toimialaratkaisut

Palvellakseen juuri tietyn kohderyhmän asiakaskuntaa on Solteqin toiminta jaettu toimialakohtaisiin kokonaisratkaisuihin, asiakaskohtaiseen järjestelmäkehitykseen sekä kauppiasvetoisille myymälöille ja erikoistavarakaupoille (Point-of-sale) suunnattuihin myymäläratkaisuihin.

Toimialakohtaisia kokonaisratkaisuja Solteq tarjoaa ketjuuntuneelle kaupalle, autokaupalle ja valikoiduille teollisuussegmenteille. Näillä toimialoilla Solteq tarjoaa ratkaisun kaikkiin asiakasyrityksen tiedonhallintaan kohdistuvissa kehitystarpeissa. (Solteq Oyj:n Internet sivut 2004.)

2.04 Solteq Merx -toiminnanohjausjärjestelmä

Solteq Merx toiminnanohjausjärjestelmä on tukku- ja vähittäiskaupan tarpeisiin suunniteltu toiminnanohjausjärjestelmä. Järjestelmä osasovelluksineen kattaa yrityksen koko materiaalihallinnon osa-alueet. Järjestelmän osa-alueet ovat myynti, osto ja varastotoiminnot. Näitä tukevia osia ovat järjestelmän ohjaus sekä tuotteiden-, asiakkaiden- ja toimittajien hallintatyökalut. Osajärjestelmiä täydentävät erilaiset liittymät muihin järjestelmiin. Merxissä on useita valmiita liittymiä yrityksen muihin sovelluksiin, kuten kassa- ja taloushallintojärjestelmiin. Merxin käyttöä tukevat myös monipuoliset selailutoiminnot, järjestelmän yhteisestä relaatiotietokannasta. Merx toimii iSeries laitealustalla jonka käyttöjärjestelmä on OS/400.

Järjestelmän käyttö on rakennettu joustavaksi, jotta erityyppiset yritykset pystyvät käyttämään sitä omissa liiketoiminnoissaan. Joustavuus ilmenee mm. siten, että järjestelmä palvelee moniyritys- ja monivarasto-ominaisuuksia hyödyntäviä yrityksiä yhtä hyvin, kuten yhden varaston yrityksiäkin. Järjestelmästä löytyvät myös erityyppiset myyntitoiminnot, joista järjestelmä ylläpitää monipuolista tilastomateriaalia. Järjestelmän ostotoiminnot mahdollistavat manuaaliset tai automaattiset ostot. Merxin varastotoiminnot tukevat useita erilaisia toimintatapoja, varastokohtaisesti. (Solteq Merx -esite 2003.)

2.05 Solteq ja XML

Solteq toimittaa ratkaisuja erilaisille toimialoille ja erilaisiin ympäristöihin. XML:ää käytetään useimmissa ratkaisuissa kuten Java ja Microsoft .Net teknologioihin perustuvissa sovelluksissa. SAP järjestelmissä käytetään tiedonsiirtojen rakentamissa tiettyä XML:ään pohjautuvaa tekniikkaa.

Solteq Merxissä XML:ää ei ole paljoa vielä käytetty, koska aikaisemmin on puuttunut tarvittavat työkalut sen käsittelemiseen. Ylipäätään iSeries laitealustalle ja OS/400 käyttöjärjestelmän Solteqissa käytettyihin työkaluihin ei XML -ominaisuuksia ole ollut saatavissa. Tällä hetkellä asiakkailta tulee paljon kysyntää XML toiminnoille ja tarvittavat työkalutkin XML -dokumenttien käsittelyyn ovat saatavilla, joten toimintojen kehittäminen on tullut ajankohtaiseksi myös Merxin osalta.

3 Tärkeitä käsitteitä

3.01 IBM iSeries -laitealusta

IBM iSeries sarjan tuotteet ovat integroituja palvelinkokonaisuuksia, joissa on useiden käyttöjärjestelmäympäristöjen yhtäaikaisen käytön tuki. IBM iSeries palvelimet ovat suunniteltu yksinkertaistamaan järjestelmäympäristöjä ja nopeuttamaan sähköisen liiketoiminnan käyttöönottoa. Erityisiä ominaisuuksia palvelimissa ovat käyttövarmuus ja hallinnointi tarpeen vähäisyys. IBM iSeries palvelimet tunnettiin aikaisemmin nimellä AS/400. (IBM:n Internet sivut 1994 - 2005.)

3.02 OS/400 käyttöjärjestelmä

IBM iSeries laitealustan peruskäyttöjärjestelmä on OS/400. OS/400 muodostaa integroidun kokonaisuuden IBM DB2 tietokannan kanssa. OS/400 on suunniteltu suuria tietomääriä käsittelevien sovelluksien alustaksi. Käyttöjärjestelmän tämän hetkinen versio on VR5R3. Yhdessä iSeries palvelinten kanssa käyttöjärjestelmä muodostaa iSeries järjestelmän. (IBM:n Internet sivut 1994 - 2005.)

3.03 Integrated Language Environment

Integrated Language Environment eli ILE käsittää joukon työkaluja ja toimintoja, jotka ovat suunniteltu parantamaan sovelluskehitystä iSeries järjestelmässä. Nämä toiminnot antavat mahdollisuuden tehdä modulaarisia sovelluksia, joissa on rinnakkaisesti suoritettavia eri kielillä ohjelmoituja moduuleita. ILE tarjoaa ohjelmointiin myös erilaisia sisään rakennettuja funktioita, jotka mahdollistavat mm. dynaamisen muistinkäsittelyn ja päivämäärien käsittelyn.

ILE:n merkitys iSeries järjestelmän ohjelmistokehityksessä on merkittävä, sillä sen avulla yksinkertaiset työkalut saavat paljon toimivia lisäominaisuuksia ja järjestelmän perinteiset lauserakenteisiin ohjelmointikieliin saadaan nykyaikaisia kehittyneitä piirteitä. (IBM:n Internet sivut 1994 – 2005. ILE RPG Reference Version 5.)

3.04 ILE RPG -ohjelmointikieli

Määritelmä

ILE RPG on johdannainen RPG IV ohjelmointikielestä ILE -ympäristössä. Syntaksiltaan ja ominaisuuksiltaan se vastaa edeltäjäänsä. Käytännössä ILE RPG on RPG IV:tä, johon on lisätty ILE:n tarjoamat lisäominaisuudet.

Ohjelmointi

RPG IV:n ollessa täysin lomakepohjainen kieli tarjoaa ILE RPG mahdollisuuden vapaamuotoiseen syntaksiin. Solteqin käytännön mukaisesti tässä tutkintotyössä käytetään kuitenkin lomakepohjaista syntaksia. Lomakepohjaisuudella tarkoitetaan sitä, että kaikilla kielen toiminnoilla on määrätyt paikkansa. Kun ohjelmoija haluaa tehdä ehdollisen lausekkeen, täytyy IF -avainsana kirjoittaa ruudulle juuri oikeaan lomakerivin positioon. Näiden lomakerivien tulee olla ohjelmakoodissa määrätyssä järjestyksessä kääntäjää varten.
(ILE RPG Reference Version 5.)

Rivien järjestys lomakepohjaisessa RPG IV ja ILERPG syntaksissa:

1. Tiedostot
2. Tiedostojen kenttien ja nimien ohjelman sisäiset muutokset
3. Muuttuja määrittelyt, tietorakenteet (ohjelman sisäiset tiedostot), taulukot ja ohjelman aliproseduurien prototyypit (esittelyt)
4. Parametrilistat
5. Avainlistat
6. Ohjelman logiikka (I/O)
7. Ohjelman lopetus

Lisäksi ohjelmakoodissa lopetuksen jälkeen voi olla seuraavia toimintoja:

1. Alirutiineja
2. Tietuekäsittelyjä
3. Aliproceduureja
4. Listauskuvauksia

Ominaisuudet

RPG IV:n verrattuna ILE RPG on monipuolisempi ja modernimpi ohjelmointikieli. Kuitenkin kieli on rakenteeltaan yksinkertaista ja soveltuu ainoastaan OS/400 käyttöjärjestelmään. ILE RPG, kuten RPG IV:kin, soveltuu parhaiten suurten tietomäärien käsittelyyn, koska siinä on sisään rakennettuna tehokkaat tietokannan käsittely toiminnot. Esimerkiksi tiedostojen avaaminen ja sulkeminen on täysin automaattista.

Ohjelmoijalle ILE RPG on yksinkertaisuutensa vuoksi helppoa opetella ja ymmärtää.

Rajoitukset

Vaikka ILE:n myötä RPG on saanut paljon uusia ominaisuuksia, sen heikkoudet ovat edelleen samat kuin vahvuudetkin. Siinä missä yksinkertainen rakenne ja valmiit tietokantaoperaatiot tuovat tehokkuutta perinteisiin iSeries järjestelmän sovelluksiin niin monimutkaisempien rakenteiden ja toimintojen luominen hankaloituu. Esimerkiksi tietovirtatiedostojen tuki puuttuu. (Salminen Jan ja Saviahde Timo 2003 ILE RPG -niksejä /ILE RPG FAQ. Timo Saviahde 2003 ILE RPG koulutusopas RPG:n osaajille. ILE RPG Reference Version 5.)

3.05 XML

Määritelmä

XML (Extensible MarkUp Language) on metakieli, eli sillä kuvataan tiedon rakennetta. Tieto eli data koostuu yleensä tekstistä tai kuvista, jotka on järjestetty tietyn periaatteen mukaisesti. XML on väline, jolla voidaan esittää tiedon hierarkia tai kapseloida data.

Rakenne

Reilusti yleistettynä XML:ää voisi kuvata itse luotuna merkintäkielenä, johon voidaan luoda rakenne kielioppi tarpeen mukaan. XML -muotoisen tiedon esitystapaa sanotaan XML -dokumentiksi. Dokumentti koostuu elementeistä, jotka sisältävät varsinaisen datan. Elementeillä voi myös olla ominaisuuksia eli attribuutteja. XML -dokumentilla voi olla myös omia määritteitään kuten merkistötunnus ja versionumero.

XML:n syntaksi:

```
<?xml version="1.0" encoding="UTF-8"?>
<dokumentti>
        <elementti attribuutti = ”ominaisuus” > Data
</Elementti>
```

Kuva 1. XML:n syntaksi.

Esitysmalli

XML -dokumentin elementit ovat osa ns. dokumenttipuuta. Dokumenttipuu on esitysmalli, jonka avulla asiakirjan rakenteellisuus esitetään ja tulkitaan. DOM (document object model) -oliomallin mukaan XML -dokumentti on hierarkkinen puutietorakenne, joka koostuu etukäteen rajoittamasta määrästä solmuja (nodes). Jokainen solmu on yksittäinen olio hierarkiassa. Dokumenttiolio on koko hierarkian ylimmällä tasolla. Jokaisella solmulla voi olla useita lapsisolmuja (child nodes, children). Vastaavasti kaikilla solmuilla on paitsi juurisolmulla (body, root node) on äitisolmu (parent node).

Dokumenttipuun rakenne muistuttaa hyvin pitkälti oikeaa puuta:

1. Puussa on juuri; dokumenttipuussa on runko (body, root node)

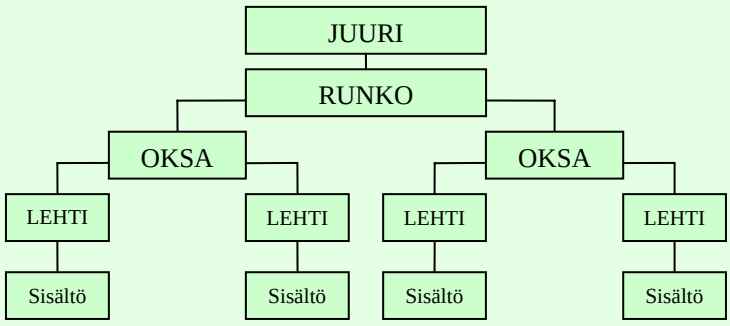
2. Puussa on oksia; dokumenttipuussa on lapsielementtejä (children, child node)
3. Puussa on lehtiä; dokumenttipuussa on lehtiä (nodes), joilla ei ole enää omia lapsielementtejä.

Esimerkki XML –dokumentista:

```
<?xml version="1.0" encoding="UTF-8"?>
<JUURI>
    <RUNKO>
        <OKSA>
            <LEHTI> Sisältö1</LEHTI>
            <LEHTI> Sisältö2</LEHTI>
        </OKSA>
        <OKSA>
            <LEHTI> Sisältö3</LEHTI>
            <LEHTI> Sisältö4</LEHTI>
        </OKSA>
    </RUNKO>
</JUURI>
```

Kuva 2. Esimerkki XML -dokumentista.

Dokumenttipuun rakenne:



Kuva 3. Dokumenttipuun rakenne.

Tärkein piirre dokumenttipuussa on, että siinä voidaan liikkua vapaasti ja se voidaan käydä läpi viittamaalla mihin tahansa solmun lapseen, vanhempaan tai sisarukseen. Jokainen solmu, eli dokumentin elementti, on oma olionsa, jolla on omat ominaisuudet ja metodit. (Koulutus ja konsultointipalvelu KK Mediat 2000 - 2004.)

Rakennekuvaukset

XML -dokumentin rakenne voidaan kuvata, joko dokumentin yhteydessä DTD (document type definition) -kuvauksella tai ulkoisella XML

-schema tiedostolla. DTD -kuvaus voidaan esittää myös ulkoisessa tiedostossa. Tiedon rakenteen kuvaamista tarvitaan, jos XML -dokumentti halutaan validoida. Validoiminen tarkoittaa sekä XML -dokumentin rakenteen että tietosisällön tarkastamista dokumentin kuvauksen mukaisesti. Tästä hyödytään tiedonsiirroissa, koska esim. järjestelmään saapuva XML -dokumentti voidaan näin tarkastaa ennen käsittelyä. Validoinnissa esiintyvät virheet voidaan huomioida ohjelmakoodissa ja näin ollen vähennetään riskiä tietojen korruptoitumisesta siirron aikana.

DTD -kuvaus

Yleisimmin käytetty tekniikka XML -dokumenttien rakenteen kuvaamiseen on DTD kuvaukset. DTD kuvaukset sisältävät elementtien hierarkian sekä elementtien nimen ja sisällön muodon.

Esimerkki DTD kuvauksesta (kuvaa em. XML –dokumentin rakenteen):

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT JUURI (RUNKO)>
<!ELEMENT LEHTI (#PCDATA)>
<!ELEMENT OKSA (LEHTI+)>
```

Kuva 4. Esimerkki DTD -kuvauksesta.

Ulkoisen DTD –kuvauksen liittäminen XML –dokumenttiin:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE JUURI SYSTEM "DTD_tiedoston_nimi.dtd">
<JUURI> ...
```

Kuva 5. Ulkoisen DTD -kuvauksen liittäminen XML -dokumenttiin.

XML -schema

XML -scheman avulla dokumenttien rakenne voidaan määritellä tarkemmalla tasolla. Nimiavaruuksien hyödyntäminen, muissa määrityksissä käytettävien kantaluokkien rakentamisen mahdollisuus ja tietotyyppien erittely antavat etua varsinkin tiedonsiirroissa, joissa tietojen eheyden säilyttäminen ja tietotyyppien ymmärtäminen ovat keskeisiä asioita. Nykypäivänä onkin suositeltavaa käyttää schemoja XML -dokumenttien rakenteen kuvauksissa. (Helsingin ammattikorkeakoulun internet sivut 2005.)

Esimerkki schema –kuvauksesta (kuvaa em. XML –dokumentin rakenteen):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="JUURI">
 <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="RUNKO" maxOccurs="1">
       <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="OKSA" type="xsd:string"/>
          <xsd:element name="LEHTI"
maxOccurs="unbounded" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Kuva 6. Esimerkki schema -kuvauksesta.

Schema –kuvauksen liittäminen XML –dokumenttiin:

```
<?xml version="1.0" encoding="UTF-8"?>
<JUURI xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="schema_tiedoston
nimi.xsd">
        <RUNKO> ...
```

Kuva 7. Schema -kuvauksen liittäminen XML-dokumenttiin.

3.06 XML -parseri

Määritelmä

XML on joustava, avoin ja voimakas tapa vaihtaa tietoa erilaisten laitealustojen, ohjelmistojen, maantieteellisten rajojen ja liiketoimintojen välillä. XML -muotoisen tiedon muodostamiseen ja käsittelemiseen ohjelmallisesti tarvitaan XML -parseri. Parserin, eli jäsentimen, toiminta perustuu XML:n standardoituun rakenteeseen ja kielioppiin. Juuri tämä yhtenäinen käsittelytapa mahdollistaa XML:n käytön eri ympäristöissä.

Parserin avulla tapahtuu XML:n luominen, liikkuminen dokumenttipuussa ja tiedon muuttaminen. XML -parseri koostuu joukosta eri toimintoihin tarkoitetuista ohjelmointirajapinnoista (API). (IBM:n internet sivut 1994 - 2005.)

Parserin toiminta

Yksinkertaisimmillaan parseri "vain" lukee rakenteen ja muodostaa siitä dokumenttipuun, mutta käytännössä useimmat parserit tekevät myös muototarkistuksen. Useimmille parsereille riittää se, että dokumentti on oikeamuotoinen (well-formed), kun taas jotkut hallitsevat myös DTD:n ja/tai Scheman mukaan validoinnin. Tarpeesta riippuu, kuinka perusteellista oikeellisuuden tarkastelua kannattaa tehdä. XML:n tulkitsemiseen liittyy se tiedon siirron kannalta tärkeä piirre, että käsittelevän ohjelman on helppo hylätä virheellinen tiedosto, jos parserin muototarkastus ei onnistu. Myös virheen muoto saadaan helposti selville. Tämän vuoksi XML:n jäsentämisessä syntyy vähemmän tulkintavirheitä, mikä johtaa yhtenäisempiin ja selkeämpiin toteutuksiin. (Jouni Heikniemen Internet sivut 2001.)

3.07 XML Toolkit for iSeries

XML Toolkit for iSeries on IBM:n tuote, joka tarjoaa XML -parserirajapinnan iSeries laitealustan ohjelmointikielille, kuten Javalle, C++:lle ja proseduraalisille ohjelmointikielille ILE RPG, ILE COBOL ja ILE C. Kaikki nämä parserirajapinnat perustuvat järjestelmä riippumattomaan Apache Open Source lähdekoodiin, joka noudattaa World Wide Web Consortium (W3C) XML -standardeja. (IBM:n Internet sivut 1994 - 2005.)

3.08 EDI/OVT

Määritelmä

EDI/OVT tarkoittaa elektronista, määrämuotoista ja automaattista tiedonsiirtoa yritysten sekä julkishallinnon tietojärjestelmien välillä. Sen tuomia etuja kuten; paperirutiinien vähentämistä, nopeutta ja virheettömyyttä voidaan hyödyntää kaikissa yrityksen toiminnoissa. Lyhenne EDI tulee englannin kielen sanoista Electronic Data Interchange ja OVT on EDI:n suomenkielinen vastine eli organisaatioiden välinen tiedonsiirto.

OVT:llä tarkoitetaan menettelyä, jossa tietojärjestelmän A tiedoista muodostetaan määrämuotoinen tietovirta, joka lähetetään sähköisenä tiedonsiirtona tietojärjestelmään B. Järjestelmässä B tietovirta puretaan ohjelmallisesti suoraan järjestelmän tietoihin, joista tiedot ovat heti järjestelmän käytettävissä. Esimerkkejä tietovirroista ovat tilaus, lasku, laskun maksuosoitus pankkiin, hinnasto tai tuoteluettelo.

Hyödyt

Sähköisen tiedonsiirron hyötyjä ovat tiedon nopea kulku ja virheettömyys. Vastaanottaja saa tiedon edelleen käytettävässä muodossa ja tiedonhallinta tehostuu. Automatisointi tuo suuria kustannussäästöjä. Täy-

sin manuaalisesti tietokoneella tehty lasku maksaa 10 - 25 euroa riippuen laskujen määrästä. Sama aineisto täysin automatisoituna (EDI) maksaa alle euron/lasku. Pankkisiirrossa, jossa turvavaatimukset ovat suuret, mutta tiedot vähäiset, kustannukset ovat pienimmillään 15 senttiä. (Tuomas Sippalan Internet sivut 2003 - 2005.)

Rakenne

EDI on kolmen eri moduulin – tietosisällön, esitystavan ja tiedon siirtämisen – muodostama kokonaisuus. Koska kaikki kolme moduulia ovat täysin erillisiä kokonaisuuksia, voidaan niitä kehittää toisistaan riippumatta.

Tietosisältö on se EDI:n osa-alue, jota kehitetään eniten. Tietosisältö tarkoittaa tietyn tyyppisen siirrettävän tiedon sisältöä. Esimerkiksi laskutiedoilla ja tuotetiedoilla on omia tietosisältömäärityksiä eri tarkoituksiin ja toimintatapoihin. Laskutietojen tietosisältö tarkoittaa sitä, mitä tietoja laskusta välitetään organisaatioiden välillä.

EDI:n esitystapa tarkoittaa tiedon muotoa siirron aikana. EDI pyrkii tarjoamaan yhden yhtäläisen tietojärjestelmien esitystavoista riippumattoman käytännön organisaatioiden välisiin tiedonsiirtoihin. EDI:n esitystavoista vakiintunein ja tunnetuin on EDIFACT -standardi (Electronic Data Interchange For Administration, Commerce and Transport), joka nimestään huolimatta on käytössä kaikilla liiketoiminnan alueilla. (Jarna Hakala, EDI - Electronical Data Interchange /OVT – Organisaatioiden välinen tiedonsiirto 3.11.1998.)

Standardit

XML -tiedonsiirto, jossa siirrettävän tiedon kuvaus noudattaa jotain tiettyä standardia, toteuttaa EDI:n määritelmän, joten XML voidaan lukea yhdeksi EDI:n esitystavaksi, joka tarjoaa hyvien ominaisuuksiensa ansiosta varteen otettavan vaihtoehdon perinteisille esitystavoille. Kuitenkaan tänä päivänä XML -tiedonsiirtoja ei vielä mielletä EDI:ksi, koska käytössä olevat XML -standardit eivät ole vielä vakiintuneita. EDI:llä tarkoitetaankin lähinnä EDIFACT:iin tai muihin standardeihin perustuvia tiedonsiirtoja. (Jarna Hakala, EDI - Electronical Data Interchange /OVT – Organisaatioiden välinen tiedonsiirto 3.11.1998.)

Tietoliikenne

Tiedon fyysiseen siirtämiseen on tarjolla useita eri tapoja. Seuraavassa on lueteltu sanomien kuljetuksessa käytettäviä vaihtoehtoja:

- X.400-sanomanvälitysjärjestelmä
- BSC/RJE (IBM ympäristössä käytetty etätöiden siirtomenettely)
- TCP/IP FTP (Transmission Control Protocol/Internet Protocol File Transfer Protocol, TCP/IP-protokollaperheen tiedostonsiirtoprotokolla)
- OFTP (ODETTE File Transfer Protocol)
- Useiden palveluverkkojen EDI tai aineistonsiirtopalvelut
- Lähes kaikki menettelyt, joiden avulla voidaan siirtää tiedostoja

(Jarna Hakala, EDI - Electronical Data Interchange /OVT – Organisaatioiden välinen tiedonsiirto 3.11.1998.)

4 EDI/OVT liittymät Solteq Merxissä

Liittymätyypit

Solteq Merxiin on rakennettu, sen kehityskaaren aikana, kymmeniä erilaisia liittymiä. Osa näistä liittymistä on ns. villejä yksittäistapauksia, joissa liittymän rakenne ei perustu mihinkään EDI -standardiin. EDI -tiedonsiirroiksi voidaan määritellä esim. erilaiset myynti- ja ostotilaus liittymät, jotka noudattavat jotain yleisesti tunnettua standardia liittymäkuvausta. Muita EDI -liittymiä on olemassa mm. tuotetietojen, hinnastojen, tilausvahvistusten ja laskujen siirtoon.

Liittymätekniikat

Perinteisesti Merxin liittymissä käytetty tekniikka on sellainen, jossa siirrettävä tieto poimitaan tietokannasta määrämuotoiseen peräkkäistiedostoon. Siirrettävä sanoma muodostuu tästä tiedostosta, joka lähetetään FTP protokollaa hyväksi käyttäen kohdejärjestelmään. Vastaavasti myös järjestelmään saapuvat tiedot otetaan vastaan samantyyppiseen ”inhouse -tiedostoon”, kuin lähetyksessä muodostetaan.

Esimerkiksi toimittajatiedot poimitaan tai vastaan otetaan tiedostoon, joka tietueessa on yksi 200 alfanumeerista merkkiä pitkä kenttä. Kentässä tietyt positiot on varattu tietyille tiedoille, jotka määritellään liittymäkuvauksessa.

Esimerkki tiedonsiirroissa käytetystä tietojen esitystavasta:

Tietue: (Alkupositio……Loppupositio)

1……
……200

Toimittajan numero 7 numeerista merkkiä 0 desimaalia:

1…7

Toimittajan nimi 20 alfanumeerista merkkiä:

9……29

Toimittajan nimen jatke 20 alfanumeerista merkkiä:

30……50

Toimittajan postiosoite 15 Alfanumeerista merkkiä:

52…67 … jne.

Kuva 8. Esimerkki tiedonsiirroissa käytetystä tietojen esitystavasta.

Käytetyn tekniikassa siirrettävä massa on tiedon määrän nähden optimaalinen, koska tiedot siirretään aina mahdollisimman suppeassa muodossa. Tällöin siirrettävän datan määrä pienenee, joten nopeus kasvaa. Siirtotiedoston arkistointiin ei myöskään siis tarvita ylimääräistä tilaa. Myös käytetyn ohjelmointikielen, eli RPG:n, ominaisuudet riittävät tämän tyyppisen tiedon käsittelyyn.

Tietojen käsittely

Tietojen käsittely tässä muodossa on kuitenkin hankalaa. Varsinkin numeerisen ja alfanumeerisen tiedon erittely on vaikeaa. Käytännössä jokainen saapuva numeerinen tieto pitää tutkia sisältääkö se vain sallittuja merkkejä, koska iSeries järjestelmä ei pysty käsittelemään väärämuotoista tietoa. Jos tutkittu merkkijono sisältää vain numeerisia merkkejä, se voidaan konvertoida varsinaisesti numeeriseen muotoon. Tässä pitää huomioida myös mahdollinen etunollien puuttuminen.

Rajoitukset

Nykyaikaisessa tietojenkäsittelyssä toimintojen vaatimukset ja määrittelyt voivat muuttua nopeassa tahdissa. Tämä aiheuttaa edellä mainitun tyyppisten liittymien ylläpidolle ongelmia, koska käytetyssä ohjelmointitekniikassa siirrettävien tietojen muuttaminen tai lisääminen edellyttää aina monimutkaisia ohjelmamuutoksia.

EDIFACT

EDIFACT -muunninta ei Solteq Merxissä ole, vaan kyseinen ominaisuus on liitettävissä järjestelmään omalla ohjelmointirajapinnallaan. Muunnin tarkoittaa sovellusta, joka muuntaa tietokantatiedoston määriteltyyn EDIFACT -esitysmuotoon. Tämä toiminto on Solteqin yhteistyökumppanin jokaiselle yksittäiselle asiakkaalle erikseen toimittama ja tämä menettely aiheuttaa ylläpito-ongelmia. EDIFACT -standardiin perustuvat tiedonsiirrotkin siis kuuluvat järjestelmän tiedonsiirtovalikoimaan.

Yhteenveto

Suurin ongelma olemassa olevien liittymien kohdalla on kuitenkin niiden kirjavuus. Esimerkiksi tuotetiedolle on tehty monia erilaisia liittymiä, jotka kaikki pohjautuvat erilaisiin sanomastandardeihin, vaikka tuotetiedot sinällään ovat aina samankaltaisia. Liittymien hallintaa kokonaisuutena ei tämän vuoksi voida tehdä. Tämä aiheuttaa kustannuksia ja resurssien turhakäyttöä, koska joudutaan useasti keksimään pyörä uudelleen.

5 XML – tiedonsiirtomuotona

XML:ään liitetään yleensä kolme uskomusta:

1. Tiedon avoimuus ja vapaus
2. Universaali tietojenkäsittelyratkaisu
3. Teknologian määräämä toimintamalli

1. XML itsessään ei tee tietojen sisältöä vapaaksi tai avoimeksi, sillä se on vain työkalu, jolla kuvataan tiedon rakennetta. Kuitenkin XML on vapaasti sovellettavissa oleva työkalu, joka voi lisätä tiedon avoimuutta siten, että yritys A:n tuottamat XML -dokumentit voidaan käsitellä yritys B:n järjestelmässä.

2. Jos kaikille tiedoille määritettäisiin maailmanlaajuisesti yhtenäinen standardi XML -schema, niin sen avulla voitaisiin ratkaista kaikki tiedonsiirtojen ongelmat. Kuitenkaan tämä ei ole nykypäivänä mahdollista, joten XML -tiedonsiirroissa on tiedoille usein määritettävä schema

liittymäkohtaisesti. Esimerkiksi valtakunnan tasolla on tietyille tiedoille jo olemassa standardeja esitystapoja. Tästä hyvänä esimerkkinä ovat pankkiyhdistyksen laatimat FINVOICE laskustandardit, jotka perustuvat XML -schemaan.

3. Kolmannen uskomuksen tarkempi määritys voisi olla se, että yhteiskunnan osien käyttämät tietojärjestelmät muokattaisiin XML -teknologian mukaisesti. Esimerkiksi tuotetiedoille määritettäisiin yhtenäinen XML -schema, ja kaikki järjestelmät tunnistaisivat kyseisen scheman mukaisia tietoja. Käytännössä tietojärjestelmät ja XML -teknologia mallintavat yhteiskuntaa ja XML antaa ihmisille mahdollisuuden tehdä muutoksia vallitseviin toimintatapoihin ja käytäntöihin. (Jari Multisilta 5.11.2001. Mihin XML soveltuu: XML:n hyödyt ja haitat)

Tietojärjestelmien välinen tiedonvaihto ja erityisesti järjestelmien integraatio ovat eräitä tietojenkäsittelyn perinteisempiä ongelmia. Tiedonvaihdossa kaksi järjestelmää toimii yhdessä siten, että tiedon lähettäjä järjestelmässä muodostetaan siirtotiedosto, jonka vastaan ottava järjestelmä pystyy tulkitsemaan. Tässä käytetään apuna yhdessä sovittuja menetelmiä, kuten tiedon rakenne ja muoto sekä käytettävä tietoliikenne menetelmä. Kuinka XML sitten soveltuu järjestelmien väliseen integraatioon sen eri osa-alueille? (Timo Helander, XML käyttäminen sovellusten välisessä tiedonvaihdossa.)

5.01 Ominaisuudet

5.01.1 Tiedon ymmärrettävyys

Järjestelmien välisessä tiedonsiirrossa siirrettävän tiedon ymmärrettävyydellä on suuri merkitys. Siirrettävän tiedon tulee olla muodoltaan sellaista, että lähettävä osapuoli pystyy sitä muodostamaan ja vastaanottava osapuoli pystyy sitä käsittelemään. Ongelmia tähän tuovat laitealustojen ja ohjelmistojen eroavaisuudet, kuten Unix (Linux) – PC tai PC – iSeries ja merkistöeroavaisuudet, kuten skandinaaviset merkit ja aasialaiset merkistöt.

XML:n parhaita puolia on se, että tänä päivänä kaikille laitealustoille ja käyttöjärjestelmille löytyy työkalut sen käsittelemiseen. Tämän mahdollistaa se, että XML on puhdasta tekstitietoa, jota kaikki järjestelmät pystyvät tulkitsemaan.

Merkistöongelmat XML:ssä on pyritty ratkaisemaan määrittämällä dokumenteille niissä käytettävä merkistötunnus. Käytettävä XML -parseri ratkaisee mitä merkistöjä voidaan käyttää. Käytännössä tämä ei ole aivan ongelmatonta, sillä esimerkiksi iSerieksen XML Toolkitin ILE RPG:n XML -parseri käyttää oletuksena pohjoisamerikkalaista UTF-8 merkistöä ja tämän takia kaikki dokumentissa käytettävä tieto on muu-

tettava erikseen länsieurooppalaiseksi ISO-8859-1 merkistöksi suomalaisissa järjestelmissä.

5.01.2 Tiedon rakenne

Tiedonsiirroissa käytettävän siirtotiedoston rakenne aiheuttaa usein siirroissa tapahtuvia virheitä. Siirtotiedoston eli siirrettävän tiedon rakenne täytyy olla yksiselitteistä ja vakiomuotoista. Siirtojen yleisimmät virheet syntyvät usein näiden ominaisuuksien huonosta suunnittelusta tai toteutuksesta. Nykypäivän nopeasti muuttuvassa maailmassa myös siirrettävien tietojen vaatimukset voivat muuttua usein. Tämä edellyttää tiedon rakenteelta joustavuutta, koska yksittäisen tiedon lisääminen tai poistaminen ei saa aiheuttaa koko liittymän uudelleen rakentamista.

XML on parhaimmillaan juuri tässä. Hyvin suunniteltu ja toteutettu XML -siirtotiedoston schema kuvaus, eli tietojen rakenteen kuvaus antaa mahdollisuuden vakioida siirrettävien tietojen rakenne. XML:n mahdollistama tietojen hierarkian kuvaus parantaa tiedon yksiselitteisyyttä. Siirtojen virheet vähenevät, kun näitä ominaisuuksia käytetään sillä XML -parseri tekee tarvittaessa muototarkastuksen sekä lähettävässä, että vastaanottavassa järjestelmässä. XML -scheman avulla voidaan myös tehdä tiedon esitystavoille standardeja, jolloin jokaiseen liittymään ei tarvitse tehdä kokonaan omia ohjelmia, sillä voidaan käyttää valmiita luokkia ja proseduureja. Tämä lisää tietojen vapautta ja käytettävyyttä.

Joustavuutensa ansiosta XML helpottaa myös muutosten tekemistä olemassa oleviin tiedonsiirtoihin. Riittää, kun tehdään muutos siirtotiedoston kuvaukseen ja lisätään käsitteleviin ohjelmiin kyseisen tiedon muodostus tai tallennus. Tämän mahdollistaa XML -tietojen hierarkkisuus ja kapselointi.

5.01.3 Tiedon siirtäminen

Tehokkaassa tiedonsiirrossa tietoliikenteen välityksellä tapahtuva siirto vaatii suurien tiedostojen osalla isoa siirtokapasiteettia. Tehokkainta olisi välittää tietoja järjestelmästä toiseen mahdollisimman tiiviissä muodossa.

XML -tiedostoa ei voida pitää kovin tiiviinä pakettina, koska se sisältää tieto-osan lisäksi myös tiedon rakenteen. XML -tiedoston koko voi olla itse asiassa yli puolet enemmän mitä pelkän tieto-osan koko on. XML:n käyttäminen tiedonsiirroissa vaatiikin hyviä tietoliikenneyhteyksiä. Tänä päivänä siirtonopeuksien kasvaessa myös XML:n käyttömahdollisuudet ovat lisääntymässä.

5.01.4 Tiedon käsittely

Pelkkä tiedon lähettäminen ja vastaanottaminen ei vielä anna paljoa hyötyä liittymästä. Tärkeää on nimenomaan se kuinka tietoa käsitellään ja mihin sitä käytetään. Tieto pitää pystyä käsittelemään mahdollisim-

man pitkälle koneellisesti, jotta liittymästä saadaan tehokkuus- ja kustannushyötyä asianomaisille yrityksille.

Esimerkkinä tiedon tehokkaasta käytöstä voidaan käyttää yrityksen laskutusjärjestelmää, jossa halutaan lähettää yritysasiakkaille laskut sähköisessä muodossa ja yksityisasiakkaille paperimuodossa. Lisäksi yritys haluaa arkistoida lähetetyt laskut paperimuodossa. Myynnistä muodostuneista laskutiedoista muodostetaan koneellisesti siirtotiedosto, joka lähetetään OVT:na tulostus ja jatkolähetys palvelun tarjoajan järjestelmään. Palvelun tarjoajan järjestelmä erottelee tulostettavat ja sähköisesti lähetettävät laskut toisistaan ja postittaa tai jatko lähettää ne laskutusosoitteisiin. Tällä tavoin tavarat myynyt yritys säästää kustannuksissa ulkoistamalla laskun käsittelyn ja saa tehokkuutta toimintaansa koneellisen käsittelyn ansioista.

Edellä mainittu toimintapa on juuri tätä päivää monissa yrityksissä. Kuinka sitten XML:ää voidaan tässä hyödyntää ja miksi siirtotiedoston pitäisi olla XML -muotoinen? Vastaus kysymykseen löytyy XML:n ominaisuuksista. Laskutietojen siirtämisessä voidaan käyttää Suomen pankkiyhdistyksen standardoimalla FINVOICE XML -schemalla kuvattua siirtotiedostoa. Tämä mahdollistaa sen, että samalla ohjelmalla voidaan vastaanottaa ja lähettää laskutietoja monesta paikasta. XML mahdollistaa siis erilaisten tietojen kuvaustapojen yhtenäistämisen. XML:n hierarkkisuus ja tietojen kapselointi mahdollistaa usean laskun lähettämisen kerralla ja samalla vastaanottava ohjelma osaa helposti erotella yksittäiset laskut toisistaan. XML:n esittämistapojen monimuotoisuutta voidaan hyödyntää siinä, että laskuja tulostettaessa tai ruudulta katseltaessa voidaan sama XML -tiedosto esittää XSL tai XSLT käännöksen avulla ruudulla HTML:nä tai PDF muodossa paperille tulostettuna.

XML -tiedoston muodostaminen ja käsittely vaatii tietojärjestelmältä paljon tehoa ja muistia. Tämä johtuu XML -parserin toiminnasta, jossa XML -dokumentti tallennetaan koneen keskusmuistiin puutietorakenteen mukaiseen muotoon. Puussa liikkuminen ja varsinkin muototarkastus vaatii paljon prosessoritehoa ja keskusmuistia. Vaikka nykyään teknologian kehittyessä koneiden suorituskyky kasvaa, niin silti XML:ää käsittelevissä ohjelmissa tulee välttää edestakaista liikkumissa dokumenttipuun solmuissa.

5.02 Liittymän rakentaminen

Tänä päivänä lähes kaikki järjestelmätoimittajat pystyvät rakentamaan XML -liittymiä, mutta kaikkia XML:n ominaisuuksia ei osata aina hyödyntää. Tämä johtuu siitä, että XML on vielä nuori teknologia ja XML:n käyttö tiedonsiirtomuotona asettaa liittymän suunnitteluun ja toteutukseen joitakin vaatimuksia. Kuinka sitten XML:n käytöstä saa-

daan parhaiten hyödyt esille ja mitä asioita pitää huomioida liittymää rakennettaessa.

5.02.1 Järjestelmävaatimukset

XML -muotoisen tiedon siirto järjestelmien välillä vaatii kummaltakin liittymän osapuolelta valmiutta käsitellä XML:ää. Käytännössä tämä tarkoittaa sitä, että järjestelmissä on oltava XML -parseri, joka pystyy käsittelemään siirrettävää tiedostoa. XML:ää on pystyttävä usein myös ohjelmallisesti käsittelemään, vaikka järjestelmien tietokantasovelluksissa olisikin valmius muodostaa ja vastaanottaa XML -muotoista tietoa, sillä tietoja voidaan joutua täydentämään tai muokkaamaan ennen lähetystä tai tallennusta. XML -dokumentin muodostaminen tai lukeminen vaatii XML -parserilta DOM tai SAX rajapinnan. Esimerkiksi järjestelmässä A voi olla käytössä 10 numeerista merkkiä pitkä tuotenumero ja järjestelmässä B voi olla käytössä 20 alfanumeerista merkkiä pitkä tuotenumero. Tällöin jommassakummassa päässä pitää tehdä muunnos kyseiselle tiedolle.

XML:n käsittely ohjelmassa taas vaatii järjestelmiltä riittävän suurta tehoa ja suurten XML -dokumenttien siirtäminen edellyttää hyviä tietoliikenneyhteyksiä. Nykyään lähes kaikilla järjestelmätoimittajilla on valmiudet tuottaa XML -liittymiä, mutta ennen liittymän tekemisen aloittamista kannattaa tehdä selvitys kaikkien osapuolten osalta.

5.02.2 Liittymän suunnittelu

Sen jälkeen, kun osapuolet ovat selvittäneet valmiudet toteuttaa XML -tiedonsiirto, tärkein asia on siirtotiedoston kuvauksen suunnittelu. Siirtotiedoston kuvausta varten on selvitettävä mitä tietoja siirretään. Tämän jälkeen kannattaa selvittää onko kyseisten tietojen siirtoon olemassa valmista standardoitua kuvausta. Joistakin tiedoista on olemassa useita standardeja, joten niistä kannattaa valita sopivin. Valmiiksi standardoidussa kuvauksessa on se hyvä puoli, että siihen ei tule niin usein muutoksia ja kuvaus on usein myös hyvin suunniteltu. Hyvä esimerkkinä standardoidusta siirtokuvauksesta on edellä mainittu FINVOICE laskutietojen siirtokuvaus. Standardista siirtotiedoston kuvauksesta tekee hyvän sen yleisyys ja rakenne. FINVOICE on Suomessa yleisin laskutietojen kuvaus menetelmä, joten sen käyttö on suositeltavaa. Tämä lisää liittymän uudelleen käytettävyyttä, jos tehdään samantyyppinen liittymä useaan järjestelmään. Jos kuvaukseen tulee muutos, niin sama muutos pätee sitten kaikkiin järjestelmiin.

Aina ei siirrettäville tiedoille löydy sopivaa valmista kuvausta. Tällöin kuvaus pitää suunnitella yhdessä liittymän toisen osapuolen kanssa ja sovittava tarkasti kuinka menetellään, jos kuvausta halutaan muuttaa. Kuvauksessa kannattaa käyttää aina XML -schema määritystä, jos se on mahdollista, koska se on paljon tarkempi kuin DTD kuvaus. Siirtotiedoston kuvaus pitää olla riittävän tarkasti määritelty, mutta liiallisuuksiinkaan ei kannata mennä. Esimerkiksi elementtien tietotyyppien

kuvaus kannattaa yleensä jättää pois, koska käsittelevät järjestelmät pystyvät tekemään tiedoille tarvittaessa tyyppimuunnoksia. Hyvä kuvaus on kompromissi kahden järjestelmän välillä, joka tarjoaa riittävästi tietoa ilman turhaa rönsyilyä ja jossa tietojen hierarkia ja kapselointi mallintaa oikeaa elämää.

Nykyään on kuitenkin olemassa paljon liittymiä, joissa siirtotiedostolle ei ole tehty minkäänlaista kuvausta, jolloin rakenteesta ja siirrettävistä tiedoista on sovittu muilla tavoin. Tätä toimintatapaa on perusteltu sillä, että näin on ennenkin toimittu perinteisissä liittymissä, joissa kuvausta ei voida tehdä. Kuvauksen puuttuminen aiheuttaa sen, että siirrettävää tiedostoa ei voida tarkastaa ennen lähetystä tai vastaanottoa ja siten menetetään yksi XML:n suurimmista eduista.

5.02.3 Liittymän toteutus

Liittymän rakentamisessa XML:n ominaisuuksien hyödyntäminen edellyttää niiden tuntemista. Mitä liittymän toteutuksessa on huomioitava, jotta lopputulos olisi hyvä?

XML:ää käsittelevissä ohjelmissa on syytä pyrkiä mahdollisimman suureen tehokkuuteen, koska parserin toiminta vaatii paljon laiteresursseja. XML:ää tietokantaan purkavissa ohjelmissa pitää välttää turhaa liikkumista dokumenttipuun solmuissa. Hyväksi tavaksi on osoittautunut dokumenttipuun solmujen viittauksien lukeminen listatietorakenteeseen. Solmulistan järjestys perustuu dokumenttipuun hierarkiaan. Ohjelmalogiikka perustuu tällöin siihen, että jokaisessa solmussa käydään vain kerran, kun sen tieto otetaan talteen. XML:n muodostuksessa tehokkuutta voidaan parantaa lisäämällä solmuja dokumenttipuuhun vallitsevan hierarkian mukaisessa järjestyksessä. Käsiteltävän XML -tiedoston rakenne ei kuitenkaan aina mahdollista kyseisen logiikan käyttöä. Yleisin poikkeuksen aiheuttaja on tilanne, jossa käsiteltävän solmun vanhempia ei tiedetä. Tällöin on pakko etsiä solmulle oikea paikka käymällä läpi dokumenttipuuta solmu solmulta.

XML -liittymissä voidaan vähentää siirrossa tapahtuvien virheiden määrää, käyttämällä parserin muototarkastusta eli validointia. Jos käytössä on siirtotiedoston kuvaus, niin ensimmäinen askel käsittelevässä ohjelmassa on tiedoston validointi, jossa käsiteltävän tiedoston rakenne ja sisältö tarkastetaan kuvausta vastaan. Tällöin siirronaikana tapahtuneet tietojen korruptoitumiset tai muodostuksessa tapahtuneet virheet saadaan selville ennen tietojen varsinaista käsittelyä. Vastaavasti siirtotiedostoa muodostettaessa validointi kannattaa tehdä viimeiseksi ennen tiedoston lähetystä. Käytettäessä XML -schemaa siirtotiedoston kuvauksena ei siirrossa voi tapahtua muotovirheitä, jos lähetyksessä ja vastaanotossa on tehty validointi.
XML:n uudelleen käytettävyys kannattaa huomioida tiedoston käsittelyssä siten, että käsiteltyä tiedostoa ei välttämättä kannata heti poistaa järjestelmästä. Esimerkiksi XML -muotoinen lasku voidaan tulostaa

ruudulle tai paperille myöhemmin XSL tai XSLT käännöksenä. Jos siirrettävän aineiston koko on suuri, kannattaa tiedosto kuitenkin tiivistää pakattuun muotoon säilytystä varten.

Suurten siirtoaineistojen kohdalla muodostettava tiedosto voidaan tehdä tiivistettyyn muotoon jättämällä ns. white space, eli elementtien väliset tyhjät tilat pois. Tästä on hyötyä sillä suurin osa FTP -protokollaa hyödyntävistä tiedonsiirtosovelluksista käsittelee nämä tilat tyhjinä merkkeinä, jolloin tiedoston koko kasvaa. Useimmissa parsereissa voidaan ohjata tätä ominaisuutta validoinnin yhteydessä. Parserin toiminnan kannalta tällä ei ole merkitystä, joten toimintoa kannattaa käyttää siirron nopeuden lisäämiseksi

Esimerkki XML-dokumentista, jossa white spacet ovat näkyvissä:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BATCH>
        <INVOICE>
          <Lang>FI</Lang>
          <InvType Type="IN"></InvType>
          <InvNr>30</InvNr>
          <Header>LASKU/FAKTURA</Header>
          <InvDate>20031104</InvDate>
          <DueDate>20031104</DueDate>
        </INVOICE>
</BATCH>
```

Kuva 9. Esimerkki XML -dokumentista, jossa white spacet ovat näkyvissä.

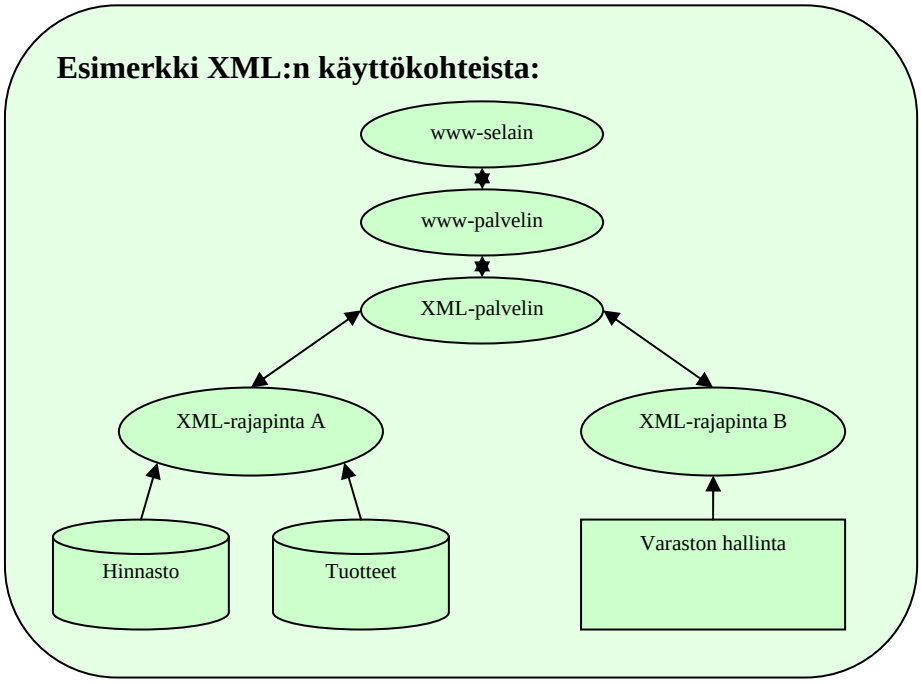
Esimerkki XML-dokumentista, josta white spacet ovat poistettu:

```
<?xml version="1.0" encoding="ISO-8859-
1"?><BATCH><INVOICE> <Lang>FI</Lang><InvType
Type="IN"></InvType><InvNr>30</InvNr>
<Header>LASKU/FAKTURA</Header><InvDate>20031104</In
vDate><DueDate>20031104</DueDate></INVOICE></BATCH
```

Kuva 10. Esimerkki XML -dokumentista, josta white spacet ovat poistettu.

Koska XML noudattaa aina tiettyjä kielioppisääntöjä, voidaan sen käsittelyä varten tehdä valmiita funktioita, luokkia tai proseduureja riippuen käytettävästä ohjelmointikielestä. Näitä yleiskäyttöisiä toiminnal-

lisuuksia voidaan siten käyttää useissa kohteissa, jolloin saadaan lisättyä tehokkuutta ja laskettua kustannuksia liittymien rakentamisessa.



Kuva 11. Esimerkki XML:n käyttökohteista.

5.02.4 Yhteenveto XML -tiedonsiirroista

XML on oikeassa paikassa, ja oikealla tavalla, käytettynä joustava ja tehokas tapa siirtää tietoa järjestelmästä toiseen. XML:ää tulisi käyttää, jos siirrettävä tieto on muodoltaan luonnollisesti rakenteellista ja tietoa halutaan muuntaa tai käyttää edelleen käsittelevässä järjestelmässä. XML:n käytön vaatimukset tulee selvittää ennen liittymän toteutuksen aloittamista. Lineaarisen ja yksinkertaisen tiedon kuvaamisessa on edelleenkin tehokkaampaa käyttää perinteisiä tiedonsiirtomuotoja. Teknologiana XML on edelleen nuori ja kehittyvä. XML:n merkitys tulevaisuuden tiedonsiirroissa tulee olemaan suurempi, mitä se on tällä hetkellä. Nykyaikaiselta toiminnanohjausjärjestelmältä vaaditaan kykyä käsitellä XML -muotoista tietoa.

6 XML ja EDI -standardit

EDI -standardit ja XML ovat tänä päivänä kilpailevia teknologioita yritysten välisissä tiedonsiirroissa. Eri toiminnanohjausjärjestelmissä on erilaisia valmiuksia käsitellä eri tietomuotoja. Tänä päivänä ei ole tavatonta, että paikasta A paikkaan B kulkeva tieto muunnetaan muodosta toiseen useaan otteeseen erivälityspalvelimissa ennen tiedon käsittelyä ja käyttämistä kohdejärjestelmässä. Tällainen menettely on tuottavaa liiketoimintaa operaattoriyrityksille, joiden toiminta perustuu tiedon muuntamiseen, säilyttämiseen ja toimintaa tukeviin lisäarvopalveluihin. Tietoa käyttäville yrityksille muuntaminen kuitenkin maksaa aikaa ja rahaa. Yksi syy nykyiseen tilanteeseen on se, että yritykset ympäri maailmaa ovat käyttäneet EDI -standardien kehitykseen paljon aikaa

rahaa ja tämän panostuksen hedelmistä halutaan nyt nauttia, eikä kaikilla tahoilla ole siten suurta halua siirtyä XML:n pariin. Kuinka EDI -standardeja ja XML:ää voidaan sitten vertailla ja onko tulevaisuus XML:ssä vai voidaanko nämä kaksi yhdistää tulevaisuudessa yhdeksi toimivaksi kokonaisuudeksi?

6.01 XML vs. EDI standardit

EDI -standardien ja XML:n vertailu on vaikeaa pelkästään niiden määrittelyjen vuoksi. Siinä missä esimerkiksi EDIFACT on puhtaasti tiedonsiirtoa järjestelmästä toiseen, ilman käyttäjän puuttumista tietoon on XML -tiedonsiirtomuodon lisäksi myös väline interaktiiviseen tietojen esittämiseen ja käsittelyyn. Uusien tiedonsiirtohankkeiden perusteena on yleensä aina yrityksen pyrkimys laskea toimitusketjusta syntyviä kustannuksia tai luoda uutta liiketoimintaa. Näin ollen XML:ää ja EDI -standardeja voidaan kuitenkin vertailla tästä näkökulmasta, koska valinta käytettävästä teknologiasta syntyy edellä mainitulla periaatteella.

6.01.1 Sähköisen kaupankäynnin standardit

Sähköisellä kaupankäynnillä pyritään tiivistämään yritysten välisiä toimintoja ja saamaan tällä aikaan kustannussäästöjä ja kilpailuetuja. Automaattiset tiedonsiirrot kuuluvat olennaisena osana tähän integroitumisprosessisiin. Yritysten kannattaa tehdä mahdollisemman monikäyttöisiä liittymärajapintoja, koska samaa tekniikka voidaan silloin käyttää hyväksi useiden yritysten välillä käytävässä tietojen vaihdossa. Tästä saadaan merkittäviä kustannussäästöjä. Jotta uusi rajapinta olisi mahdollisimman monikäyttöinen, kannattaa käyttää siirtotiedoston kuvaksia jotka ovat yleisesti tunnettuja. Tätä kuitenkin vaikeuttaa erialaisten XML -standardien kirjavuus.

Kummallakin tekniikalla toteutettu tiedonsiirto tarvitsee määritetyn kuvauksen siirrettävän tiedoston rakenteesta. Näitä kuvauksia on EDI -maailmassa tarjolla paljon ja ne ovat useasti yleisiä sekä vakiintuneita standardeja. XML -tiedonsiirtoihin on myös tarjolla paljon erilaisia rakennekuvauksia, mutta ongelmana on se, että vakiintuneita ja yleisesti käytettäviä standardi kuvauksia ei ole olemassa läheskään kaikille tiedoille. Tämä johtuu siitä, että esimerkiksi EDIFACT -standardin mukainen sanoma on aina rakenteeltaan vakio ja XML -sanoma puolestaan voi muuttua siirrettävien tietojen ja tiedoston kuvauksen (XML -schema tai DTD) puitteissa. Tähän vaikuttaa myös se, että XML on kehityskaarensa alkuvaiheessa ja vakiintuneita käytäntöjä ei ole vielä syntynyt.

6.01.2 Tiedon esitystavat ja käytettävyys

EDI -standardeissa tiedon muoto on puhtaasti tietokoneen luoma ja sitä voi käyttää ainoastaan toinen tietokone. XML -muotoista tieto voidaan siirtää koneelta koneelle, mutta tieto on myös ihmiselle ymmärrettävää.

Tämä mahdollistaan tiedon käyttämisen ilman konversiota käyttäjärapinnassa. XML:n monimuotoisuus antaa siis mahdollisuuden tiedon katselemiseen ja muokkaamiseen käyttäjän toimesta. Yleensä raaka XML -data kuitenkin käännetään XSL tai XSLT käännöksellä, joko selaimelle HTML -muotoon tai paperille tulostettavaksi PDF – muotoon.

XML:n nousemista tulevaisuudessa johtavaksi tiedonsiirtomuodoksi edesauttaa käyttöliittymien muuttuminen selainpohjaisiksi, koska verrattuna EDI -standardeihin se vähentää tiedon muokkaamistarvetta koneen ja käyttäjärajapinnan välillä. XML on siis monikäyttöinen tiedostomuoto, siinä missä esimerkiksi EDIFACT tiedon esitystapana soveltuu pelkästään tiedon siirtämiseen.

Tietojen siirtämisen kannalta XML on kuitenkin vaativampi kuin muut EDI:n esitystavat. XML -dokumenttia siirrettäessä järjestelmästä toiseen ei siirretä pelkästään tietoa, vaan myös tiedon rakenteen kuvaus. Tämä lisää siirtotiedoston fyysistä kokoa, jolloin tehokkaan toiminnan vaatimuksena on suurempi tiedonsiirtokapasiteetti. Tiedonsiirtokapasiteetin lisäys taas tarkoittaa lisääntyneitä kuluja.

6.01.3 Tiedon käsittely

XML:n ymmärrettävyys ja joustavuus antavat hyvät lähtökohdat tiedon käsittelyyn, koska se on helppo oppia ja käyttää. Lisäksi useat XML -työkalut ja parserit perustuvat avoimeen lähdekoodin ja ovat vapaasti ladattavissa internetistä. Tämä tarjoaa XML -teknologialle merkittävän kustannussäästön verrattuna kilpaileviin standardeihin. Esimerkiksi EDIFACT -tiedonsiirron rakentaminen vaatii ohjelmoijalta syvää perehtymistä standardiin ja tasokasta ohjelmointiosaamista. EDI -tiedonsiirtojen tekemiseen kehitetyt työkalut ovat kalliita kaupallisia sovelluksia, joiden käyttämisestä yritykselle aiheutuu lisenssimaksujen lisäksi useasti tapahtumakohtaisia käyttökuluja, jotka aiheutuvat kolmansien osapuolien, yleensä välityspalvelin operaattoreiden, mukana olosta.

EDI -tiedonsiirrot käyttävät useasti maksullisia VAN (Value added network) tietoverkkoja, koska yritysten on halvempaa lähettää tiedot välitys ja muuntopalvelimelle EDI -muotoon muutettavaksi, kuin rakentaa omaan järjestelmän monipuolisia EDI -muuntimia. Nämä tietoverkot pohjautuvat useimmiten transaktiopohjaiseen laskutukseen, jolloin yritys maksaa jokaisesta lähetystä tai vastaanotetusta sanomasta. VAN -tietoverkoista saatava hyöty on siinä, että sanomaa voidaan muuntaa eri muotoihin vastaanottajan perusteella. Tämä taas on merkittävä asia uutta liiketoimintaa suunniteltaessa.

XML -tiedonsiirto puolestaan ei vaadi VAN -tietoverkon käyttöä, koska XML -sanomat voidaan välittää julkista Internetiä tai VPN (Virtual Privat Network) -yhteyttä pitkin. XML:n käytön helppous houkuttaa

yrityksiä rakentamaan järjestelmiinsä rajapinnat, jotka eivät vaadi kalliiden muunnospalvelimien käyttöä.

6.01.4 Yhteenveto XML:n ja EDI -standardien vertailusta

XML:n ja EDI -standardien vertailutaulukko (J. Ricker et al 2002):

XML:

1. Optimoitu helppoon ohjelmointiin ja luettavuuteen
2. Vaati internetpalvelimen (internet palvelimen hinta voi olla 10 osa EDI -palvelimen hinnasta)
3. Pienet aloituskustannukset
4. Yksinkertainen ja luettava esitysmuoto
5. Vaatii vähemmän vaativaa ohjelmointiosaamista
6. Halvat kehitystyökalut
7. Esitysmuoto on myös ihmiselle ymmärrettävää
8. Standardit kehitysvaiheessa
9. Sanomastandardit

EDI standardit:

1. Optimoitu kokoon (XML sanoma on kooltaan 10 kertainen verrattuna EDI sanomaan)
2. Vaati EDI palvelimen
3. Korkeat aloituskustannukset
4. Monimutkaiset ja vaikeasti opittavat esitysmuodot
5. Vaatii korkeatasoista ohjelmointiosaamista
6. Kalliit kehitystyökalut
7. Esitysmuodon ymmärtää vain tietokone
8. Vakiintuneet standardit
9. Sekä Bisnesprosessistandardit, että sanomastandardit

Kuva 12. XML:n ja EDI -standardien vertailutaulukko. (J. Ricker et al 2002.)

Yllä olevasta J. Rickerin vertailutaulukosta nähdään helposti XML ja EDI - tiedonsiirtojen peruseroavaisuudet. Käytännössä valinta näiden kahden liittymätekniikan välillä kuitenkin perustuu yleensä kustannuksiin, eli halvin muoto yleensä valitaan. Jos yritys on panostanut esimerkiksi EDIFACT -tekniikkaan ja sillä on valmiit työkalut liittymän rakentamiseen, niin todennäköisesti halutaan jatkaa samalla menetelmällä. Koska EDI -standardit ovat tällä hetkellä käytetyimpiä tiedonsiirtomuotoja, voidaan päätellä, että XML ei tule niitä korvaamaan lyhyellä aikajänteellä. XML on käytetympi tiedonsiirtomuoto siellä, missä kummatkin, tai toinen tiedonsiirron osapuolista käyttää selainpohjaista tietojärjestelmää.

VAN -tietoverkkoja ja välityspalvelimia ylläpitävien operaattoriyritysten palveluiden kalleus ja selainpohjaisten järjestelmien yleistyminen ovat kuitenkin ajaneet useita yrityksiä XML:n pariin ja jatkossa tämä suuntaus näyttäisi vahvistuvan. Tämä taas tulee nopeuttamaan XML -standardien vakiintumista. Vaikka XML:n käsittely helpompaa, niin tällä hetkellä XML -osaamista on kuitenkin rajallinen määrä, eikä sen ominaisuuksia aina tiedetä tarpeeksi hyvin.

Ominaisuuksiensa vuoksi XML -tiedonsiirtomuotona palvelee tällä hetkellä parhaiten web-sovelluksia ja muita reaaliaikaisesti tapahtuvia tiedonsiirtoja, joissa siirrettävän tiedon määrä ei ole suuri ja joissa tiedon rakenteella on suuri merkitys. Suurten tietomäärien siirtämiseen EDI -standardit ovat kuitenkin tällä hetkellä toimivin tapa.

6.02 XML/EDI tulevaisuuden ratkaisu?

Määritelmä

Kuten aikaisemmin EDI/OVT osioissa todettiin, XML on toisaalta myös yksi EDI:n esitystapa. Tämä mahdollistaa näiden kahden teknologian yhdistymisen. EDI:n heikkoutena on aina pidetty sen käyttämän VAN -teknologian kalleutta ja toisaalta EDI:n parhaina puolina pidetään vakiintuneita standardeja niin liiketoiminta prosesseissa kuin sanomissakin. Tästä syystä on kehitetty käsite internet EDI, joka tarkoittaa julkisen internetin välityksellä toimivaa EDI tiedonsiirtoa. XML astuu tässä kohden kehiin EDI:n esitystavaksi, jolloin käsite muuttuu XML/EDI:ksi. XML/EDI:n parhaiten tunnettuja standardeja ovat RosettaNet ja ebXML. (Electronic Commerce Promotion Council of Japan 2003, ECOM. Internet EDI (XML/EDI) Introduction Guidebook)

Hyödyt

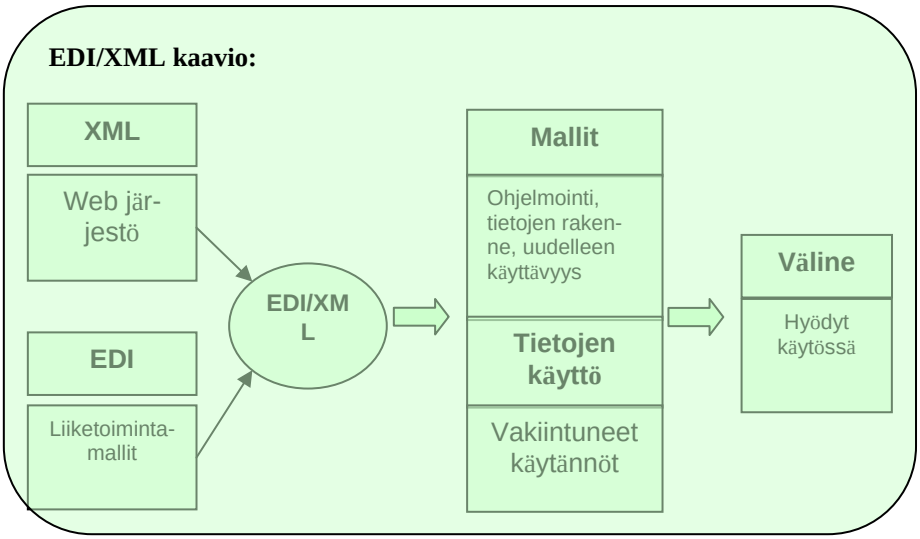
XML/EDI:ssä yhdistyvät EDI:n vakiintuneet liiketoimintaprosessien standardit ja XML:n monikäyttöisyys. Tämä tarkoittaa sitä, että yhdistelmäksi saadaan toimiva kokonaisuus, joka pohjautuu vahvoihin ja toimiviin liiketoimintamalleihin sekä nykyaikaiseen web-teknologiaan. Tässä saatavat hyödyt vaikuttavat suoraan tietojen siirrosta syntyviin kuluihin, koska siirroissa ei enää tarvita kolmannen osapuolen ylläpitämiä VAN -tietoverkkoja. Myös XML:n monikäyttöisyys ja joustavuus tuovat kustannussäästöjä, kun siirrettäviä tietoja voidaan käyttää suoraan käyttöliittymässä. Myös liiketoiminnan lisääminen on mahdollista, sillä reaaliajassa tehtävät tietojen siirrot ja kyselyt järjestelmien välillä voidaan toteuttaa kohtuullisilla kustannuksilla.

Rajoitukset

XML/EDI tulee olemaan varteenotettavampi vaihtoehto EDI:n perinteisille standardeille, mitä pelkkä XML -sanoman välitys, koska se tukee samoja liiketoimintaprosessi standardeja, kuin esim. EDIFACT. Näin ollen perinteisiin EDI -tiedonsiirtoihin panostaneilla yrityksillä on pienempi kynnys siirtyä XML -teknologian käyttöön. Kuitenkin tässäkin törmätään samaan ongelmaan, eli samaa XML:n käsittelyä tämä toimintamalli vaatii ja se taas aiheuttaa omat aloituskustannuksensa.

EDI ansaitsee oman kritiikkinsä kalleutensa ja monimutkaisuutensa vuoksi, mutta sen kehittämiseen käytetty 30 vuotta ei ole kokonaan turhaan heitettyä, koska XML -toimintoja voidaan rakentaa tämän kehityksen jatkeeksi, sen sijaan, että kaikki entinen hylättäisiin ja lähdettäisiin puhtaalta pöydältä. XML hyötyy EDI:stä saaduista kokemuksista ja se onkin perinyt siltä paljon hyviä ominaisuuksia. Kaikki merkit viittaisivat siihen, että XML/EDI:stä tulee luonnollinen jatke EDI:n aloitta-

malla polulla. Tästä paras viite on se, että lähes kaikki nykyaikaiset toiminnanohjausjärjestelmät tukevat XML:n käsittelyä ja toisaalta perustuvat tunnettuihin sähköisen kaupankäynnin standardeihin. (Alan Cook 1999. XML and EDI Lessons Learned and Baggage to Leave Behind)



Kuva 13. EDI/XML kaavio.

7 XML -toiminnot Solteq Merxissä

Solteq Merx on puhtaasti merkkipohjainen käyttöliittymä IBM iSeries suurtietokone ympäristössä, joka on ohjelmoitu RPG IV ja ILE RPG kielillä. Näissä ohjelmointikielissä ei ole tukea tietovirtatiedostojen lukemiselle/kirjoittamiselle, koska ne muodostavat integroidun kokonaisuuden DB2 tietokannan ja OS/400 käyttöjärjestelmän kanssa. Tämä ei siis ole mikään hyvä lähtökohta XML - toimintojen kehittämiselle, koska monille XML:n ominaisuuksille, kuten monikäyttöisyydelle, ei ole käyttöä. (Solteq Merx -esite 2003.)

XML -toimintojen tarve on kuitenkin syntynyt siitä, että järjestelmän halutaan toimivan integroidusti myös nykyaikaisten Web -sovellusten kanssa. XML -toiminnot tarkoittavat siis tässä tapauksessa XML -sanoman vastaanottoa ja lähetystä. Sanoman vastaanotossa halutaan purkaa saapuneen XML -tiedoston tiedot tietokantatiedostoon. Tietokantatiedostosta tiedot voidaan jatko käsitellä järjestelmässä käytettäviksi tiedoiksi. Vastaavasti sanoman lähetyksessä halutaan poimia järjestelmän tietokannasta tiedot haluttuun XML -formaattiin ja lähettää tiedosto vastaanottavaan järjestelmään.

Tulevaisuudessa järjestelmä voidaan kehittää kuitenkin olemassa olevilla teknologioillakin selainpohjaiseksi, jolloin XML:n ominaisuuksia voidaan paremmin hyödyntää. Tästä ei kuitenkaan ole olemassa mitään päätöksiä, joten sitä ei voida pitää perusteena siirtymisessä puhtaasti XML -tiedonsiirtoihin. Näin ollen XML -toiminnot tulevat olemaan yhtenä vaihtoehtona uusille tiedonsiirroille.

7.01 Lähtötilanteen selvitys

Tällä hetkellä järjestelmään on rakennettu muutamia yksittäisiä XML -sanoman vastaanotto ja lähetys sovelluksia. Näissä on osittain käytetty IBM:n XML Toolkit ohjelmointirajapintaa ja osittain taas XML -data on parsittu tekstin käsittelynä tietokantatiedostoista. Jälkimmäisellä tavalla toteutetuissa sovelluksissa on kuitenkin törmätty ylläpito-ongelmiin, sillä sovelluksien monimutkainen rakenne on huonosti muunneltavissa. Lisäksi sovellukset ovat kertakäyttöisiä, eli niitä ei ole voitu uudelleen käyttää toisissa toteutuksissa.

XML Toolkitin ohjelmointirajapinnan avulla tehdyissä sovelluksissa on tähän asti käytetty myös kertakäyttöistä toimintamallia, eli jokaista sovellusta kohden on ohjelmoitu uudet ohjelmat. XML Toolkitin ohjelmointirajapinta on kuitenkin monimutkainen kokonaisuus, joka vaatii ohjelmoijalta paljon perehtymistä kohteeseen.
(Solteq Merx 2004/2.)

XML -tiedonsiirtojen määrä ja kysyntä asiakasrajapinnassa on kuitenkin kokoajan ollut kasvussa, joten näiden sovellusten kehittäminen on vaatinut resursseja. Yksittäisen liittymien hinta on myös joissain tapauksissa osoittautunut suhteellisen korkeaksi. Näistä syistä on todettu, että kannattavinta on rakentaa järjestelmään monikäyttöinen XML -rajapinta, jota voidaan käyttää useimmissa uusissa XML -toteutuksissa.

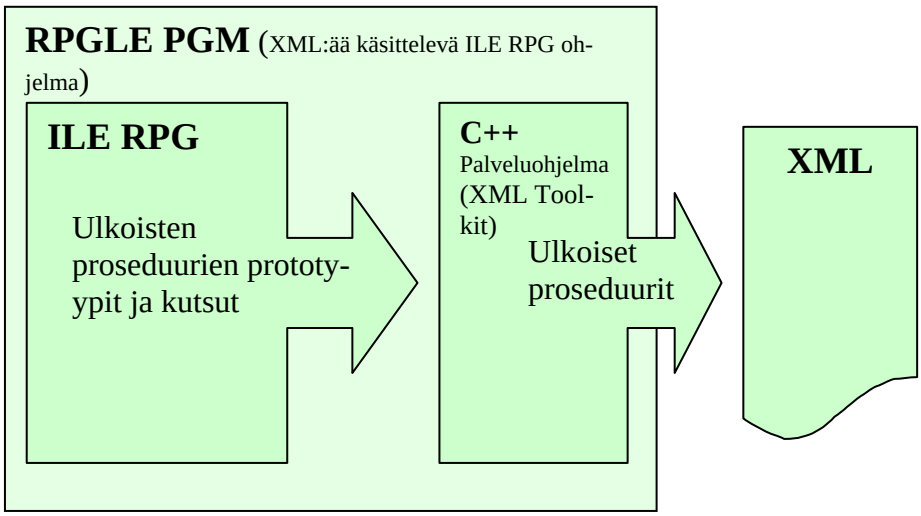
7.02 XML - toimintojen kehitys

Työkalut

XML -toimintojen kehittämisessä suurin ongelma on se, että järjestelmän ohjelmointikielissä ei ole tukea tietovirtatiedostoille. Tämä ongelma voidaan ratkaista käyttämällä XML Toolkitin ohjelmointirajapintaa, joka tarjoaa tuen XML -tiedostojen käsittelyyn tietovirtamuodossa sekä DOM että SAX mallinnuksen avulla. Lisäksi XML Toolkitin hyviä puolia ovat sen matala hinta, vaikka XML -toimintoja tarvitsevat järjestelmää käyttävät asiakasyritykset joutuvatkin siihen lisenssin hankkimaan. XML Toolkit on tällä hetkellä paras tapa käsitellä XML:ää OS/400 käyttöjärjestelmässä ILE RPG kielellä.
(Solteq Merx 2004/2. IBM:n Internet sivut 1994 - 2005.)

Ratkaisu

XML Toolkit ei sellaisenaan tarjoa rajapintaa XML -dokumentin ja tietokannan välillä, vaan se tarjoaa ohjelmointirajapinnan ILE RPG:lle XML -tiedostojen käsittelyyn. Tämä tarkoittaa sitä, että XML:n käsittelyyn on tehtävä uudet ILE RPG – ohjelmat, jotka käyttävät XML Toolkitin ohjelmointirajapintaa. Tämä rajapinta saadaan käyttöön kopioimalla (Copy Member -tekniikka) rajapinnan ulkoisten proseduurien prototyypit niitä kutsuvaan ohjelmaan ja liittämällä käännöksessä ohjelmamoduuliin palveluohjelma, jossa ulkoiset proseduurit fyysisesti ovat. Kaavio kuvassa 14 kuvaa XML:ää XML Toolkitin avulla käsittelevän ohjelman rakennetta.



Kuva 14. Kaavio XML:ää XML Toolkitin avulla käsittelevän RPG -ohjelman rakenteesta.

XML:ää käsittelevän RPGLE ohjelmaan kuuluu ILE RPG moduuli, jossa on XML Toolkitin ulkoisten proseduurien prototyypit ja näiden kutsut sekä C++ palveluohjelma, joka varsinaisesti sisältää nämä proseduurit. Varsinaisesti ulkoiset proseduurit ovat C++ palveluohjelman funktioita, jotka käsittelevät XML -dokumenttia.

XML Toolkitin myötä ILE RPG:hen saadaan tarvittava ominaisuus käsitellä XML -dokumenttia tietovirtamuodossa. Tämän vuoksi käsiteltävä dokumentti on tallennettava iSeries palvelimen IFS – hakemistorakenteeseen. IFS (Integrated File System) hakemistorakenne tarkoittaa iSeries palvelimen sisäistä tiedosto rakennetta, joka poikkeaa OS/400 käyttöjärjestelmän kirjastoihin ja tiedosto-objekteihin perustuvasta rakenteesta siten, että se vastaa toiminnallisuudeltaan Windows -käyttöjärjestelmän hakemistorakennetta.

Rajoitukset

XML Toolkit tarvitsee toimiakseen OS/400 käyttöjärjestelmästä version V5R1:sen tai uudemman. Tätä vanhemmille käyttöjärjestelmille pitää XML -tiedonsiirrot toteuttaa käyttäen, joko edellä mainittua tapaa ilman XML Toolkittiä tai käyttäen tuotteen vanhempaa versiota.

Kaikesta huolimatta XML Toolkit ei tarjoa aivan täydellistä rajapintaa XML:n käsittelyyn. Nykyinen versio tuotteesta mahdollistaa XML -tiedoston luomisen ja purkamisen, mutta esimerkiksi standardin mukaiset hakutoiminnot ja mahdollisuus integroida kaksi tiedostoa XSL -tekniikan avulla puuttuvat kokonaan. Rajapintaa kuitenkin kehitetään IBM:n toimesta jatkuvasti ja siitä saadaan säännöllisin väliajoin parannettu versio. Tällä hetkellä XML Toolkit on kuitenkin toimivin ja monipuolisin tapa käsitellä XML:ää ILE RPG kielellä ja se riittää vaatimusten mukaisen XML -rajapinnan luomiseen Solteq Merx järjestelmään.

7.03 XML – toimintojen tulevaisuus

XML -tiedonsiirtojen kysyntä tulee jatkossa olemaan tasaisen varmaa, sillä järjestelmää käyttävien yritysten sähköisen liiketoiminnan kehitys monilta osin vasta alkuvaiheessa. Tulevaisuudessa tiedon siirrot tulevat olemaan yhä olennaisempi osa yrityksien sähköistä liiketoimintaa. Tämän vuoksi on perusteltua toteuttaa järjestelmään tässä vaiheessa XML -rajapinta, jota pystytään käyttämään useissa eri hankkeissa.

Jatkossa XML:n standardien kehittyessä joudutaan varmasti tilanteeseen, jossa joudutaan päivittämään olemassa olevia XML:ää käsitteleviä ohjelmia. Tähän joudutaan myös siinä tapauksessa, että XML Toolkitin jossain tulevassa versiossa ei enää tueta tämän hetkistä rajapintaa. Siksi Solteqin tulisi olla aktiivisesti mukana kehittämässä tuotetta ja seurata XML -standardien kehitystä.

Pitemmällä tähtäimellä iSeries maailman ollessa menossa yhä enemmän Java teknologian ja sitä myötä web-käyttöliittymien suuntaan tuli myös jossain vaiheessa harkita XML:n käsittelyn siirtämistä Java -pohjaiseksi. Tämän myötä saataisiin myös käyttöön niitä ominaisuuksia XML -tiedonsiirroista, joita nykyinen järjestelmän kehitystila ei salli.

8 XML -rajapinta Solteq Merxiin

XML -rajapinnalla tarkoitetaan tässä sovelluskokonaisuutta, joka tarjoaa työkalut XML -tiedostojen vastaanottamiseen ja muodostamiseen. Rajapinnan avulla halutaan jatkossa toteuttaa järjestelmään tehtävät XML -tiedonsiirrot. Rajapinta perustuu XML Toolkitin ohjelmointirajapintaan, koska se on tällä hetkellä käyttökelpoisin tapa käsitellä XML:ää ILE RPG ohjelmissa. XML -rajapinnasta tehdään ns. lisäarvotuote, joka on liitettävissä Solteq Merxin lisäksi muihin iSeries alustalla toimiviin järjestelmiin. Hankkiessaan tämän lisäarvotuotteen asiakasyritys joutuu myös hankkimaan XML Toolkitin käyttöoikeuden. Tämän tutkintotyön myötä Solteqin tarjoama lisäarvo tässä kokonaisuudessa on nimenomaan rajapinta XML:n tietokannan välillä.

Rajapinnan toteutusta ei ole tehty omana tuotekehitysprojektinaan, vaan se perustuu Solteqissa meneillään oleviin asiakasprojekteihin. Tämä aiheuttaa sen, että kokonaisuudelle ei ole olemassa omaa dokumentoitua ja itsenäistä vaatimusmäärittelyä, vaan vaatimusmäärittelyt ovat syntyneet toiminnallisuuden osalta näistä asiakasprojekteista lisättynä vaatimuksella monikäyttöisyydestä.

8.01 Vaatimusmäärittely

XML -rajapinnan toteutuksen vaatimuksena olivat seuraavat toiminnallisuudet:

- XML:n käsittelyssä käytetään IBM:n XML Toolkitin ohjelmointirajapintaa ILE PRG -ohjelmointikielelle:
    - Käytettävä versio rajapinnasta on 5.0
- Rajapinnan avulla on pystyttävä muodostamaan XML -tiedosto:
    - XML -tiedosto muodostetaan tiedoista, jotka on poimittu tietokantatiedostoon erillisessä poimintaohjelmassa
    - Tiedot poimitaan XML:n rakenteen mukaiseen järjestykseen
    - Muodostettavan XML -tiedoston rakenteen on noudatettava poimitun aineiston rakennetta
    - Muodostettavan XML -tiedoston rakenne ja sisältö voivat muuttua
    - Tarvittaessa muodostettu tiedosto pitää pystyä validoimaan, joko DTD kuvauksen tai XML -scheman avulla ennen lähetystä:
        - Validointia ei kuitenkaan aina haluta tehdä
    - Tiedoston tietojen rakenne tai sisältö ei saa muuttua siirrettäessä tietokannasta XML -dokumenttiin
    - Muodostettava XML -dokumentti voi sisältää suuriakin tietomääriä
- Rajapinnan avulla on pystyttävä purkamaan XML -tiedosto tietokantaan:
    - XML -tiedoston tiedot puretaan tietokantatiedostoon, jossa niiden rakenne ja järjestys eivät saa muuttua
    - Purettavan XML -tiedoston rakenne ja sisältö voivat muuttua
    - Tarvittaessa vastaanotettava XML -tiedosto pitää pystyä validoimaan, joko DTD -kuvaksen ja XML -scheman avulla:
        - Validointi ei ole kuitenkaan pakollista
    - Tiedoston tietojen rakenne tai sisältö ei saa muuttua siirrettäessä XML -dokumentista tietokantaan.
    - Vastaanotettava tiedosto voi sisältää suuriakin tietomääriä
- Kokonaisuus pitää olla ylläpidettävissä
- Uusien XML -liittymien toteutus pitää olla helppoa
- Uusien XML -liittymien toteutuksissa ei tarvita XML Toolkit osaamista

8.01.1 XML -tiedoston muodostaminen

Vaatimusmäärittely XML -tiedoston muodostamiselle tarkoittavat sitä, että tehtävän sovelluksen on pystyttävä muodostamaan määrämuotoon poimitusta aineistosta halutun mukainen XML -tiedosto. Määrämuotoon poimittu tieto pitää olla sellaisessa muodossa, että sovellus osaa

sen perusteella muodostaa halutun rakenteen. Poimintatiedoista pitää siis ilmetä tietojen järjestys ja hierarkia.

Koska muodostettavan tiedoston rakenne määritetään poimintaohjelmassa tehtävään aineistoon, sovelluksen tulee siis noudattaa tätä rakennetta. Tätä varten on suunniteltava joustava ja toimiva muoto tälle poiminta-aineistolle.

Jos liittymässä kulkevalle XML -tiedostolle on määritelty DTD kuvaus tai XML -schema tulee sovelluksessa olla mahdollista validoida muodostettu tiedosto tätä kuvausta vastaan.

Siirrettävillä tiedolla on merkitystä vastaanottavalle osapuolelle vain, jos tiedon eheys säilyy siirrossa. Tämä tarkoittaa sitä, että siirrettävien tietojen hierarkia ja merkitys eivät saa muuttua siirrossa.

Suuria tietomääriä käsiteltäessä on sovelluksessa huomioitava tehokkuus ja tiedoston koko. Tavoitteena on siis muodostaa tiedosto mahdollisimman tehokkaita menetelmiä käyttäen. Muistin käsittelyyn on siis kiinnitettävä huomiota sovellusta ohjelmoitaessa. Muodostettavan tiedoston kokoon voidaan vaikuttaa lähinnä poistamalla turhat white spacet ennen lähetystä.

8.01.2 XML -tiedoston purkaminen

Vaatimukset XML -tiedoston purkamiselle tarkoittavat sitä, että vastaanotettu XML -tiedoston on pystyttävä purkamaan määrämuotoiseen muotoon, joka tallennetaan tietokantatiedostoon.

Tiedot tulee purkaa sellaiseen muotoon, että niitä edelleen käsittelevä ohjelma pystyy lukemaan ne ja käyttämään edelleen järjestelmässä. Muodossa tulee siis olla sellainen rakenne, että XML -tiedoston tietojen hierarkia ei muutu.

Vastaanotettava XML -tiedosto pitää pystyä tarvittaessa validoimaan joko DTD kuvauksen tai XML -scheman avulla, ennen tietojen purkua. Jos siirretty tiedosto ei ole validia, tietoja ei voida purkaa sillä tiedon eheys on silloin voinut rikkoutua.

Sovelluksen on pystyttävä vastaanottamaan myös suuria tietomääriä sisältäviä tietoja. Tämän vuoksi on sovelluksessa käytettävä tehokkaita menetelmiä XML:n käsittelyssä.

8.01.3 Kokonaisuuden ylläpidettävyys

Ylläpidettävyyden kannalta kokonaisuuden toteutuksessa on käytettävä hyvää ohjelmointitekniikkaa. Toteutuksessa on siis käytettävä modulointia ja yleisesti tunnettuja ohjelmointimenetelmiä. Myös ohjelmakoodin kommentointiin on kiinnitettävä huomioita.

8.01.4 Uusien XML -liittymien rakentaminen rajapintasovelluksen avulla

Toteutuksessa on huomioitava se, että tämä rajapintasovellus ei sinällään ole vielä riittävä tekijä uuden XML -liittymän rakentamiselle vaan se tarvitsee lisäksi ohjelmat jotka käsittelevät poimittua tai purettua tietoa määrämuotoisesta tiedostosta. Rajapintasovelluksen ja näiden ohjelmien välinen toiminta lopulta muodostaa tietojen siirron järjestelmästä järjestelmään. Määrämuotoinen tietorakenne, joka talletetaan tietokantatiedostoon, tulee siis olla sellainen, että sitä käsittelevät ohjelmiin voidaan helposti rakentaa logiikka, jolla tietoja käsitellään. Näissä käsittelevissä ohjelmissa ei siis enää käytetä XML Toolkitin toimintoja, vaan perustuvat tavalliseen ILE RPG ohjelmointiin.

8.02 Tekninen suunnittelu

Toteutuksen tekninen suunnittelu perustuu XML Toolkitin toimintojen käyttöön sekä tietojen poiminnassa ja purussa käytetyn määrämuotoisen tietorakenteen suunnitteluun. Lisäksi teknisessä suunnittelussa on otettu kantaa siihen, minkälaisiin toiminnallisiin moduuleihin kokonaisuus kannattaa jakaa.

8.02.1 Kokonaisuuden jakaminen toiminnallisiin moduuleihin

Lähtökohtaisesti kokonaisuus jakaantuu kahteen eri osa-alueeseen. Nämä osa-alueet ovat XML -tiedoston muodostus ja purkaminen. Tämän perusteella kokonaisuus kannattaa jakaa kahteen eri moduuliin.

Ensimmäinen moduuli on XML -tiedoston muodostusohjelma. Tämä ohjelma käyttää XML Toolkitin toimintoja XML -tiedoston muodostamiseen. Ohjelma noudattaa kohdassa XML -toimintojen kehitys esiteltyä rakennetta, jossa ohjelmaan liitetään XML Toolkitin tarjoama palveluohjelma, sekä sen proseduurien prototyypit. Ohjelman toiminta perustuu siihen, että XML -tiedoston elementit muodostetaan käyttäen rajapinnan tarjoamia ulkoisia proseduureja. Muodostettavan tiedoston sisältö ja rakenne luetaan määrämuotoon poimitusta tietokantatiedostosta.

Toinen moduuli on XML -tiedoston purkuohjelma. Tämä ohjelma käyttää XML Toolkitin toimintoja ja noudattaa samaa rakennetta, kuin muodostusohjelmakin. Käytössä ovat kuitenkin eri proseduurit, joilla tiedot haetaan tiedostosta. Tiedoston tiedoista muodostetaan määrämuotoon poimittua tietokantatiedostoa.

8.02.2 XML Toolkitin toimintojen käyttäminen

XML Toolkitin toimintojen käyttäminen vaatii seuraavia toimenpiteitä (IBM:n Internet sivut 1994 - 2005.)**:**

- Lisätään lähdekoodiin copymemberiksi tiedosto QXML4PR500 kirjastosta QXMLDEV500, jossa on rajapinnan prototyypit
- XML -parseriympäristön alustus

Muistinkäsittely ja XML -ympäristön lopetus:

- Ohjelmassa käytetään rajapinnan proseduurien kautta C++ palveluohjelman funktioita ja kaikissa näissä funktioissa ei ole dynaamista muistinkäsittelyä. Tämä aiheuttaa sen, että ohjelmassa on vapautettava käytetyt muistiresurssit asiaankuuluvilla proseduureilla.
- Ennen ohjelman suorituksen lopettamista on rajapinnan XML -ympäristön suoritus lopetettava tietyllä proseduurilla

Poikkeusten käsittely:

- XML Toolkitin ohjelmointirajapinnassa on kahden tyyppisiä poikkeussanomia:
  - Rajapinnan poikkeussanomat
  - Parserin poikkeussanomat
- Rajapinnan poikkeussanomat syntyvät proseduureissa mahdollisesti syntyvistä virheistä. Esimerkiksi, jos yritetään poistaa objekti, jota ei ole olemassa.
- Parserin poikkeussanomat syntyvät, kun parsittava XML -dokumentti ei ole oikeamuotoista tai se ei ole määritellyn kuvauksen mukaisesti validia.
- Poikkeussanomat otetaan vastaan tiettyyn, tätä tarkoitusta varten varattuun muistialueeseen. Sanoman paluukoodi määrittelee virheentyypin.

Merkkijonojen käsittely ja merkistötunnukset:

- XML Toolkitin perusmerkistönä on UTF-8, joka ei tue eurooppalaista ISO-8859-1 merkistöstandardin mukaisia skandinaavisia merkkejä.
- XML Toolkit kuitenkin mahdollistaa merkkimuunnoksen näiden kahden välillä. Muunnoksessa käytetään tiettyä proseduuria, jolle määritetään parametreiksi haluttu merkistö. Parametrin arvona käytetään Job_CSSID vakiota, jolloin String -objektin merkistötunnukseksi tulee työn merkistötunnus
- C++ palveluohjelma käsittelee merkkijonona String -olioina, joten merkkijonot on muunnettava tietyllä proseduurilla NULL -päätteiseksi String -olioksi.
- XML -elementtien ja attribuuttien nimet ovat aina UNICODE tekstimuotoisia. Tähän tarkoitukseen käytettävät merkkijonot on muunnettava asiankuuluvalla proseduurilla.

8.02.3 Tietojen lukeminen ja kirjoittaminen tietokantaan

Edellä mainittujen ohjelmien tarkoitus on joko muuntaa tietokannassa säilytettävä tieto XML -dokumentiksi tai purkaa XML -dokumentti tietokantaan. Tätä varten tarvitaan tietty tietorakenne, joka voidaan tallettaa tietokantaan, lukea XML -dokumentin haluttuun formaattiin ja johon voidaan tallettaa XML -dokumentista saadut tiedot rakenteen särkymättä.

Tässä tietorakenteessa pitää olla ohjelmien toimivuuden kannalta seuraavat tiedot:

- Eränumero, jolla tunnistetaan mitkä tiedot kuuluvat yhteen XML - tiedostoon.
- Järjestysnumero, joka järjestää tiedot haluttuun järjestykseen.
- Tasonumero, joka kertoo minkä elementin lapsielementti kyseinen tieto on
- Tiedon omistavan elementin nimi, joka kertoo toisella tavalla minkä elementin lapsielementti on kyseessä
- XML -elementin nimi.
- XML -elementin tyyppi, jolla erotetaan onko kyseessä element - tyyppinen tieto vai onko kyseessä jonkun elementin attribuutti
- Tieto-osa, jossa on elementin arvo

Tämä tietorakenne tallennetaan tietokantaan, jolloin tiedoston avaimeksi muodostuu:

- Eränumero
- Järjestysnumero
- Tasonumero

Tietorakenteesta muodostuu tietokantaan yksipuolinen puurakenne, joka noudattelee XML -dokumentin rakennetta.

8.03 Toteutuksen kuvaus

8.03.1 XML -dokumentin muodostus

Ohjelma RCREATEXML muodostaa kohdehakemistoon halutulle nimelle XML -tiedoston, jonka tiedot ja rakenne luetaan poimintatiedostosta.

Ohjelman parametrit (input):

- Lähetyserän numero
- Kohdehakemisto
- Muodostettavan tiedoston nimi

Ohjelman parametrit (output):

- Paluukoodi

Ohjelman käyttämät tietokantatiedostot:

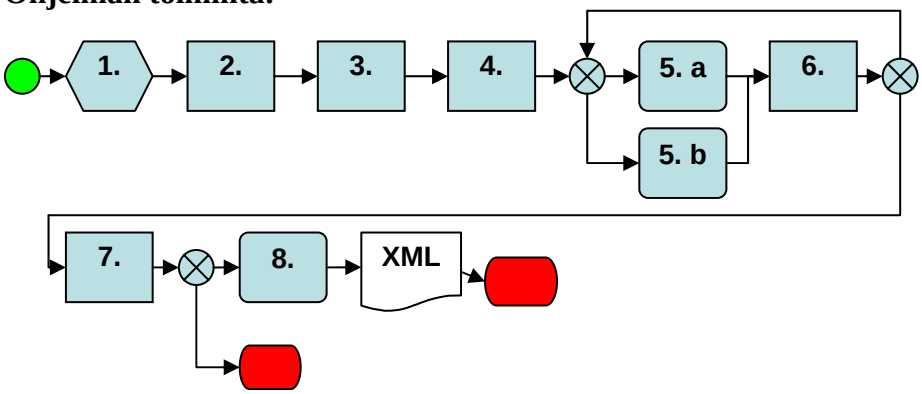
- FXMLDATA (poimintatiedosto):

```
              T I E T U E K U V A U S
TIEDOSTO FXMLDATA                    TIETUE RXMLDATA
    NIMI     PITUUS/DES TEKSTI
  1 XDNUME       7 P  0 LÄHETYSERÄN NUMERO
  2 XDJNRO       7 P  0 JÄRJESTYSNUMERO
```

```
  3 XDTASO       2 P  0 TASON NUMERO
  4 XDTANI      20 A    TASON NIMI
  5 XDELEM      50 A    XML-ELEMENTTI
  6 XDDATA     100 A    DATA-OSA
  7 XDATTR      10 A    DATA-OSA

REC.FORMAT       KEY FIELD  SEQUENCE  KEY SIGN
ZONE/DIGIT  ALT.COLL.SEQ.
               1 XDNUME        A       SIGNED
               2 XDJNRO        A       SIGNED
```

Ohjelman toiminta:



Kuva 15. XML -dokumentin muodostuksen toimintakaavio.

1. Ohjelman alustukset.
2. Luetaan poimintatiedoston ensimmäinen tietue.
3. Tehdään poimintatiedoston ensimmäisestä tietueesta XML -dokumentin juurielementti. Talletetaan elementtitaulukkoon viittaus juurielementtiin.
4. Luetaan poimintatiedoston seuraava tietue. Tutkitaan onko tietue attribuutti vai elementti.
5. A) Jos tietueen tyyppi on elementti, etsitään tietueen tasonumerolla elementtitaulukosta sen omistava elementti ja lisätään tietue sen lapsielementteihin. Lisätään uusi elementti elementtitaulukkoon tasonumeron osoittaman indeksin kohdalle.
   B) Jos tietueen tyyppi on attribuutti, lisätään se edellisen lisätyn elementin attribuutiksi.
6. Luetaan seuraava tietue poimintatiedostosta. Jos tietue löytyy, suoritetaan kohdan 5 tutkinta ja toimenpiteet. Jos kaikki tietueet on jo käsitelty, poistutaan silmukasta.
7. Tutkitaan onko XML -dokumentissa elementtejä.
8. Jos elementtejä on, tulostetaan ne tietovirtatiedostoon kohdehakemistoon.

8.03.2 XML -dokumentin purku

Ohjelma RREADXML lukee annetun XML -tiedoston tiedot ja kirjoittaa ne poiminta-aineistoksi. XML -tiedoston elementtien hierarkia säilyy myös poiminta-aineistossa.

Ohjelman parametrit (input):

- Purettavan tiedoston hakemisto ja nimi

Ohjelman parametrit (output):

- Paluukoodi

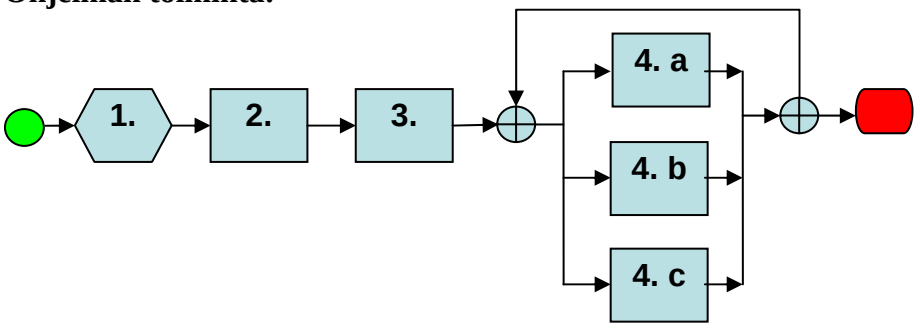
Ohjelman käyttämät tietokantatiedostot:

- FXMLDATA (poimintatiedosto):

```
          T I E T U E K U V A U S
TIEDOSTO FXMLDATA                    TIETUE RXMLDATA
    NIMI     PITUUS/DES TEKSTI
  1 XDNUME        7 P  0 LÄHETYSERÄN NUMERO
  2 XDJNRO        7 P  0 JÄRJESTYSNUMERO
  3 XDTASO        2 P  0 TASON NUMERO
  4 XDTANI       20 A    TASON NIMI
  5 XDELEM       50 A    XML-ELEMENTTI
  6 XDDATA      100 A    DATA-OSA
  7 XDATTR       10 A    DATA-OSA

REC.FORMAT        KEY FIELD  SEQUENCE  KEY SIGN
ZONE/DIGIT  ALT.COLL.SEQ.
               1 XDNUME        A       SIGNED
               2 XDJNRO        A       SIGNED
```

Ohjelman toiminta:



Kuva 16. XML -dokumentin purkamisen toimintakaavio.

1. Ohjelman alustukset
2. XML -dokumentin parsiminen. Tästä syntyy DOM puutietorakenne, jota käytetään tiedoston purkamisessa.
3. Kutsutaan rekursiivista purkuproseduuria parametrina DOM tietorakenteen ensimmäinen solmu.
4. Puretaan DOM puurakenne rekursiivisesti solmu kerrallaan. Päätellään solmun tyypistä suoritettavat toimenpiteet:
   A. Solmun tyyppi on XML -dokumentin juurielementti. Haetaan listaan ne DOM tietorakenteen solmut, jotka sisältävät juurielementin lapsielementit ja kutsutaan proseduuria uudelleen jokaiselle näistä solmuista.
   B. Solmun tyyppi on elementti. Haetaan elementin attribuutit ja otetaan ne arvoinen talteen. Jos solmun ensimmäinen lapsisolmu on tekstityyppinen, eli se sisältää elementin tieto-osan, otetaan muuttujiin talteen myös elementin tiedot. Jos ensimmäinen

solmu on elementti, kirjoitetaan elementin ja sen attribuuttien tiedot poiminta-aineistoon.

C. Solmun tyyppi on teksti. Haetaan tekstin omistavan elementin ja sen attribuuttien tiedot muuttujista sekä solmun arvo ja kirjoitetaan poiminta-aineistoon elementin tiedoiksi.

8.04 XML -liittymän rakentaminen toteutuksen avulla

Kohdassa 8.03.1 ja 8.03.2 esitellyt ohjelmakomponentit mahdollistavat XML -tiedonsiirtoon perustuvien liittymien rakentamisen. Liittymän suunnittelussa ja toteutuksessa näitä komponentteja käyttäen ei välttämättä tarvita vankkaa XML -teknologian tai XML -ohjelmoinnin tuntemusta, koska XML -muotoisen tiedon käsittely suoritetaan komponenttien sisällä. Tekijöille riittää näin ollen perusteellinen tietämys XML:n rakenteesta, koska heidän on osattava tulkita kohdassa 8.02.3 esitellyn poiminta-aineiston rakennetta.

Liittymiä rakennettaessa on huomioitava se, että jokaista erityyppistä XML -tiedonsiirtoa kohden on tehtävä omat ns. poiminta-ohjelmat, jotka käsittelevät poiminta-aineistoa. Esimerkiksi xCBL ja cXML ovat kummatkin XML -tiedonsiirron standardoituja tiedon esittämistapoja ja kummastakin standardista löytyy, lähes saman tietosisällön omaavat, tuotetietoja kuvaavat sanomamallit. Yhteneväisestä tietosisällöstä riippumatta kumpaankin liittymään tarvitaan omat poiminta-ohjelmat, koska tiedoston rakenne on erilainen.

8.04.1 XML -sanoman lähetys

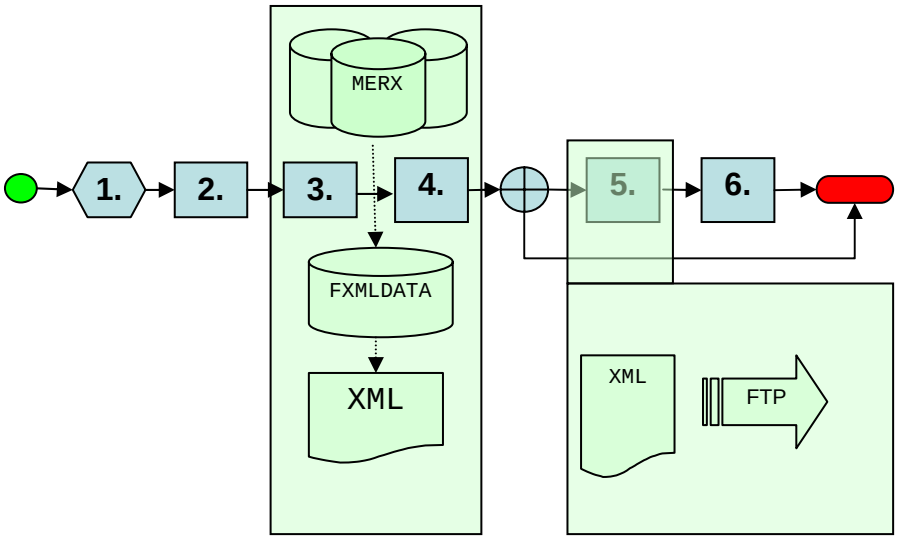
Merx järjestelmästä lähtevät tiedot ovat yleensä ostotilauksia toimittajille, tuote- ja hinnastotietoja sekä laskuja asiakkaille. Kuinka tällainen tiedonsiirto rakennetaan, jos XML:ää käytetään sanoman muotona? Mitä ohjelmakomponentteja tarvitaan ja missä järjestyksessä niitä kutsutaan? Näihin kysymyksiin pyritään vastaamaan seuraavassa selvityksessä.

Ensimmäinen tarvittava komponentti on IBM XML Toolkit, jonka suorituksen aikaisia elementtejä tarvitaan kohdassa 8.03.1 esitellyn RCREATEXML moduulin suorittamisessa. XML Toolkit lisensseineen on tämän vuoksi hankittava myös asiakasympäristöihin, joissa XML -toimintoja käytetään.

XML:n käsittelyssä käytetään siis RCREATEXML moduulia, joka lukee poimitut tiedot poiminta-aineistosta ja muodostaa niistä XML -dokumentin haluttuun ifs-hakemistoon.

Poiminta-aineiston muodostukseen tarvitaan poiminta-ohjelma, joka lukee tarvittavat tiedot tietokannasta ja kirjoittaa ne halutun XML -tiedoston rakenteen mukaisesti poimintatiedostoon.

Näistä kaikista muodostetaan yhtenäinen ajo kasaamalla ohjelmakutsut yhdeksi ohjelma kokonaisuudeksi. Tarvittaessa tähän ajoon voidaan myös lisätä automaattinen tiedoston lähettäminen halutulla tietoliikenneprotokollalla kohdejärjestelmään.



Kuva 17. XML -sanoman lähetys.

1. Ohjelman alustuksissa liitetään ohjelmaan XML Toolkit ympäristö ja XML:n käsittelymoduulit.
2. Luodaan FXMLDATA tiedostosta ajonaikainen kopio, jota käytetään poiminta-aineiston säilytyspaikkana. Kopio poistuu kokonaan ajon päättyessä, joten aineisto ei jää turhaan levylle viemään tilaa.
3. Poiminta-ohjelmassa luetaan tiedot Merxin tietokannoista ja kirjoitetaan ne poiminta-aineistoon, joka tallennetaan FXMLDATA tiedostoon.
4. RCREATEXML moduulissa luodaan poiminta-aineistosta uusi XML -tiedosto haluttuun ifs -hakemistoon.
5. Tutkitaan onnistuiko XML -tiedoston muodostus. Jos onnistui, tehdään FTP – siirto kohdejärjestelmään. Muussa tapauksessa mennään suoraan loppuun, koska tiedostoa ei muodostettu.
6. Talletetaan muodostettu ja lähetetty tiedosto tallennushakemistoon.

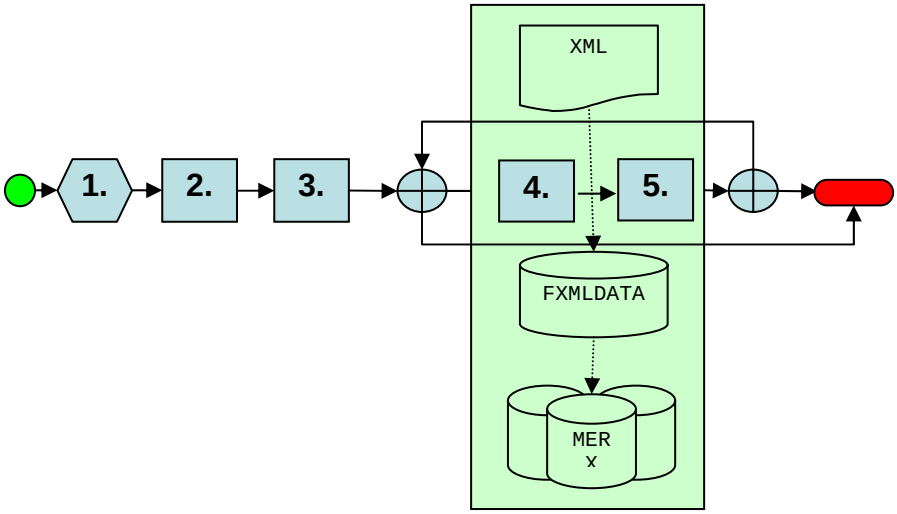
8.04.2 XML -sanoman vastaanotto

Merxiin vastaanotettavat tiedot ovat yleensä tuote- ja hinnastotietoja sekä ostolaskuja toimittajilta ja myyntitilauksia asiakkailta. Tietoja vastaanottaessa on tavoitteena saada vastaanotetut tiedot siirrettyä järjestelmän omaan tietokantaan, jotta niitä voidaan käyttää päivittäisessä liiketoiminnassa. Seuraavassa selvityksessä kerrotaan, kuinka tällaisen tiedonsiirron voi toteuttaa XML -muotoisena käyttäen kohdassa 8.03.2 esiteltyä RREADXML moduulia. Mitä komponentteja tarvitaan ja missä järjestyksessä niitä kutsutaan?

Kuten XML -sanoman muodostuksessa, vastaanotossakin tarvitaan IBM XML Toolkitin ajonaikaisia elementtejä. XML -tiedostoa käsittelevä ohjelma on RREADXML, joka lukee saapuneen tiedoston ja purkaa sen tiedot poiminta-aineistoon. Poiminta-aineiston muoto on sama kuin lähetyspuolella.

Poiminta-aineistosta tiedot kirjoitetaan Merxin tietokantoihin poimintaohjelmassa, joka lukee puretut tiedot tiedostosta FXMLDATA. FXMLDATA tiedostossa aineistossa on edelleen XML -dokumentin rakenne mukana, joten tämän avulla voidaan päätellä tietojen hierarkia.

Yleensä saapuvia tietoja ei Merx järjestelmässä kirjoiteta suoraan todellisiin tietokantoihin, vaan esimerkiksi myyntitilaussanoman tiedot kerätään vastaanottotiedostoihin, joihin sitten lisätään mm. voimassa olevat hinnat ennen kuin niistä muodostetaan varsinaisia järjestelmän myyntitilauksia.



Kuva 18. XML -sanoman vastaanotto.

1. Ohjelman alustuksissa liitetään ohjelmaan XML Toolkit ympäristö ja XML käsittelymoduulit.
2. Luodaan FXMLDATA tiedostosta ajonaikainen kopio, jota käytetään poiminta-aineiston säilytyspaikkana. Kopio poistuu kokonaan ajon päättyessä, joten aineisto ei jää turhaan levylle viemään tilaa.
3. Luetaan kohdehakemistossa olevat saapuneet XML -tiedostot. Jos tiedostoja on, niin kutsutaan purkuohjelmaa RREADXML ja poimintaohjelmaa jokaiselle tiedostolle erikseen. Tällä tavoin vältetään tietojen sekoittumista eri tiedostojen kesken.
4. Moduulissa RREADXML parsitaan saapunut XML -tiedosto. Jos XML -tiedosto on muodoltaan virheetön ja tietosisältö vastaa mahdollista DTD tai schema kuvausta, se puretaan poimintatiedostoon FXMLDATA.
5. Poimintaohjelmassa luetaan tiedot poiminta-aineistosta ja kirjoitetaan ne vastaaviin Merx tiedostoihin, jolloin ne ovat järjestelmän

käytössä jatkokäsittelyä varten. Lisäksi saapunut tiedosto tallennetaan tallennushakemistoon.

9 Lopputulos ja arviointi

Tutkintotyön aiheena oli XML -toimintojen kehittäminen Solteq Merxissä. Lähtökohtaisesti tilanne oli työn aloitusvaiheessa se, että XML -tiedonsiirrot olivat järjestelmässä kokeiluasteella ja koko XML -teknologian tuntemus järjestelmän parissa toimivan osaston parissa oli vähäistä. Merxin asiakaskunnasta alkoi kuitenkin tulla koko ajan enemmän painetta XML -toimintojen kehittämiseen, joten näin ollen aloin kehittelemään uusia ratkaisumalleja näiden asiakas projektien perusteella. Työ eteni pikkuhiljaa liittymä liittymältä teknisesti toimivampaan ja taloudellisempaan suuntaan ja jossain vaiheessa huomasin, että paras ratkaisu sekä tutkintotyön tavoitteiden että Merxin ja Solteqin kannalta oli luoda tässä työssä kuvattu Solteq XML rajapinta. Samalla keräsin myös tietoja XML:n ominaisuuksista ja tutkin teknologian sopivuutta erilaisissa tiedonsiirroissa ja vertasin niitä olemassa oleviin tiedonsiirtoratkaisuihin.

9.01 Lopputulos

Tutkintotyön päällimmäisenä tavoitteena oli luoda Solteq Merxiin valmius vastaanottaa ja lähettää XML -muotoista tietoa. Tähän tarkoitukseen valmistui Solteq XML rajapinta, jossa on mukana niin XML -tiedoston vastaanotto ja lähetysmoduulit. Tämä rajapinta perustuu IBM:n XML Toolkitiin ja mahdollistaa XML -muotoiseen tiedonsiirtoon perustuvien liittymien rakentamisen ilman syvällistä XML:n teknologian osaamista. XML toimintojen toivottiin myös kasvattavan järjestelmän arvoa.

Toisena tavoitteena tutkintotyössä oli tutkia XML:ää tiedonsiirtomuotona ja kartoittaa minkä tyyppisissä liittymissä sen käytöstä saadaan etua verrattuna perinteisiin tiedonsiirtomuotoihin. Tätä varten työssä on vertailtu XML:n ja muiden tiedonsiirtomuotojen ominaisuuksia. Tämän perusteella pystytään siten hahmottamaan missä ratkaisuissa XML:n käyttäminen on mahdollista ja milloin siitä saadaan etuja.

9.02 Lopputuloksen arviointi

Solteq XML rajapinta toteuttaa sille asetetut vaatimukset ja se on myös myyty ja otettu tuotantokäyttöön muutamalle Merx järjestelmää käyttävälle asiakkaalle sekä myös yhdelle muuta iSeries ratkaisua käyttävälle yritykselle. Näin ollen tutkintotyön toteutuksen osalta voidaan sanoa, että se saavuttaa asetetut tavoitteet niin toiminnallisuuden, kuin järjestelmän arvon lisäämisen kannalta. Lisähyötynä on saatu mahdollisuus tarjota tätä ratkaisua myös muita järjestelmiä käyttäville asiakkaille.

Tässä on kuitenkin huomioitava se, että tämä ratkaisu on koko ajan edelleen kehittymässä parempaan suuntaan varsinkin tehokkuus mie-

lessä. Uusin XML muodostusohjelman versio mahdollistaa entistä suurempien aineistojen käsittelyn ja luomisen ajoaikojen pienennettyä. Tutkintotyössä luotu pohja ja kerätyt kokemukset ovat kuitenkin tärkeässä asemassa jatkokehityksen kannalta.

XML -tietämys Merxin parissa on myös huomattavasti lisääntynyt tutkintotyön kehityksen aikana ja tämän on mahdollistanut XML:n ominaisuuksien ja soveltuvuuden tutkiminen, joka kuului yhtenä osana tähän työhön. Tietämystä on pyritty lisäämään tiedottamalla ja asioista keskustelemalla. Tällä hetkellä organisaatiossa tiedetään missä XML -liittymiä voidaan käyttää ja mitä hyötyjä niistä saadaan. Kuitenkin tämän työn valmistumisen jälkeen tulee vielä kerätä kaikki tiedot yhteen ja lisätä tiedottamista tämän osalta.

Loppuraportin kannalta tuotekehitysprojekti tutkintotyönä oli hankala. XML -toimintojen kehittämiseen mennyt aika aiheutti sen, että raportti on kirjoitettu kymmenillä eri kerroilla. Aikaa kului etenkin siihen, että Solteqissa uusia ominaisuuksia kehitetään asiakasprojektien kautta ja niitä ei aina ollut tähän työhön liittyen tarjolla. Alkuvaiheessa kaksi projektia keskeytyi ulkopuolisista syistä, joten tavoitteiden mukaisen lopputuloksen syntyminen oli mahdollista vasta keväällä 2005. Lukuisilla kirjoituskerroilla on varmasti vaikutuksensa raportin kirjalliseen ulkoasuun.

Lähteet

Organisaatioiden välinen tiedonsiirto – Sähköisen kaupankäynnin seminaari 30.9.1999 [online] [Viitattu 18.4.2005]
http://www.helsinki.fi/~amertani/ovt.htm

Solteq Oyj:n Internet sivut. [online] [viitattu 12.6.2004]
www.solteq.com

Solteq vuosikertomus 2004 - suomi. 2005 Solteq Oyj.

Solteq Merx -esite [online] [viitattu 10.6.2004]
www.solteq.com

IBM:n Internet sivut 1994 – 2005 [online] [viitattu 10.11.2004 ja 2.3.2005]
www.ibm.com

ILE RPG Reference, Version 5. IBM corporation 2003-2005.

Salminen Jan ja Saviahde Timo 2003, ILE RPG -niksejä /ILE RPG faq. Solteq Oyj. Tampere.

Saviahde Timo 2003, ILE RPG koulutusopas RPG:n osaajille. Solteq Oyj. Tampere.

Koulutus ja konsultointipalvelu KK Mediat 2000 – 2004 Internet sivut 2004. [online] [viitattu kesäkuussa 2004]
http://www.2kmediat.com/kkmediat/www_2kmediatcom.asp

Helsingin ammattikorkeakoulun Internet sivut 2005. XML –SCHEMA eli skeema. [online] [viitattu 18.4.2005]
http://myy.helia.fi/~vanvu/xmlkurssi/schema.html

Jouni Heikniemen Internet sivut 2001. Mikä on XML? 19.2.2001. [online] [viitattu 14.4.2005]
http://www.heikniemi.net/kirj/moxml.html

Tuomas Sippalan Internet sivut 2003 – 2005. EDI/OVT. [online] [viitattu 1.9.2004]
http://www.kotinet.com/tuomas.sippala/ovt.htm

Jarna Hakala, EDI - Electronical Data Interchange /OVT – Organisaatioiden välinen tiedonsiirto 3.11.1998. [online] [viitattu 5.11.2004]
http://www.tml.hut.fi/Studies/Tik-110.300/1998/Essays/edi.html

Prof. Jari Multisilta 5.11.2001,Porin korkeakouluyksikkö, Mihin XML soveltuu: XML:n hyödyt ja haitat. [online] [viitattu joulukuussa 2004]

http://www.urova.fi/home/atk2001/esitykset/multisilta.pdf

Timo Helander, XML:n käyttäminen sovellusten välisessä tiedonvaihdossa. [online] [viitattu joulukuussa 2004] http://www.tml.hut.fi/Studies/Tik-110.300/1998/Essays/xml_2.html

J. Ricker et al 2002. XML:n ja EDI -standardien vertailutaulukko.

Electronic Commerce Promotion Council of Japan 2003, ECOM. Internet EDI (XML/EDI) Introduction Guidebook. [online] [viitattu marraskuussa 2004] http://www.ecom.jp/ecom_e/press/20030529/InternetEDIGuidebook.pdf

Alan Cook 1999. XML.COM. XML and EDI Lessons Learned and Baggage to Leave Behind. [online] [viitattu 13.9.2004] http://www.xml.com/pub/a/1999/08/edi/index.html?wwwrrr_19990804.txt

Solteq Merx 2004/2 toiminnanohjausjärjestelmä. Solteq Oyj.