SAVONIA

Tuotannonseurantajärjestelmän tiedonkeruukirjaston rakentaminen

Janne Koistinen

Opinnäytetyö

___. ___. _____ _____

Ammattikorkeakoulututkinto

SAVONIA-AMMATTIKORKEAKOULU **OPINNÄYTETYÖ**
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Janne Koistinen	
Työn nimi Tuotannonseurantajärjestelmän tiedonkeruukirjaston rakentaminen	
Päiväys 21.11.2011	Sivumäärä/Liitteet 33
Ohjaaja(t) lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) Enfo Oyj, suunnittelupäällikkö Antti Normaja, järjestelmäsuunnittelija Jani Heikkinen	

Tiivistelmä

Opinnäytetyön aiheena oli suunnitella ja luoda tietoa keräävä luokkakirjasto, joka keräisi tietoa erilaisista prosesseista ja välittäisi ne eteenpäin. Työ tehtiin Enfo Oyj:lle Tiedonvälityspalvelut-yksikössä osana projektitiimiä. Tiedonkeruukirjaston tarkoituksena oli saada se liitettyä Enfon järjestelmiin, joista se keräisi tietoja. Tiedot saatuaan kirjasto muodostaisi niistä tuotannonseurantajärjestelmälle luettavan sanoman ja lähettäisi sen eteenpäin.

Työ aloitettiin perehtymällä uuden tuotannonseurantajärjestelmän toimintaan sekä Enfon aineistojen käsittelyprosessien eri vaiheisiin. Kirjaston suunnittelu alkoi projektidokumentaatioiden malliviestien pohjalta, jonka jälkeen suunniteltiin viestien runko sekä yleinen toimintaperiaate. Työ sisälsi suunnittelu-, määrittely-, toteutus-, testaus- sekä integrointivaiheen.

Kirjaston teossa käytettiin apuna UltraEdit-sovelluseditoria ja sovellus ohjelmoitiin käyttäen Perl-ohjelmointikieltä. Keskeisenä osana työn teossa oli olio-ohjelmointi ja suunnittelumallit helppokäyttöisyyden hakemiseksi.

Työn lopputuloksena saatiin määritysten mukainen tiedonkeruukirjasto. Tiedonkeruukirjasto liitettiin onnistuneesti tuotannonseurantajärjestelmään.

Avainsanat
tuotannonseuranta, PERL, olio-ohjelmointi, skripti

SAVONIA UNIVERSITY OF APPLIED SCIENCES **THESIS**
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Janne Koistinen			
Title of Thesis Building Data Collecting Library for the Production Monitoring System			
Date	21 November 2011	Pages/Appendices	33
Supervisor(s) Mr Jussi Koistinen, Lecturer and Mr Sami Lahti, Lecturer			
Client Organisation /Partners Enfo Oyj, Mr Antti Normaja, Design Manager and Mr Jani Heikkinen, System Designer			

Abstract

The main goal of this final project was to design and implement an information gathering class library which would gather data from various processes and would relay them forward. The final year project was made for Enfo Oyj's Information Logistics services unit as a member of the project team. The purpose of the data collecting library was to integrate it into Enfo's systems from which it gathers the data. Once the data has been gathered, the library forms a valid message for the production monitoring system and sends it forward.

The project was started by studying the basic functions of the new production monitoring system and all the stages in Enfo's material handling processes. The designing of the library was started on the basis of the model messages which are described in the project documentation. Model messages made it possible to design the message template and general functionality of the data collecting library. The work included such stages as: planning, defining, implementation, testing and integration.

The tools which were used while making the library were UltraEdit text editor and the coding was made with the Perl programming language. Main focuses in the final year project were object oriented programming and pursue for simple usage.

As the final result of the final year project there was a fully functioning data collecting library. The library was successfully integrated into the production monitoring system.

Keywords
Production monitoring, PERL, object oriented programming, script

Alkusanat

Tämän opinnäytetyö tehtiin Enfo Oyj:lle syksyllä 2011. Työtä ohjasivat lehtori Jussi Koistinen Savonia-ammattikorkeakoulusta ja järjestelmäsuunnittelija Jani Heikkinen Enfo Oyj:stä. Haluan kiittää työn ohjaajia sekä muita projektiryhmäläisiä, jotka osallistuivat ohjaamiseen ja mahdollistivat työn valmistumisen.

Kuopiossa 21.11.2011

Janne Koistinen

SISÄLTÖ

1 JOHDANTO 9
2 PROJEKTI 11
2.1 Yleiskuvaus 11
2.2 Henkilöstö ja roolit 11
2.3 Aikataulut 11
3 KÄYTETYT TEKNIIKAT 13
3.1 Perl 13
3.2 Ultraedit 13
4 TUOTANNONSEURANTAJÄRJESTELMÄ 15
4.1 Kuvaus 15
4.2 Toiminta 16
4.3 Sanomat 17
5 TIEDONKERUUKIRJASTO 19
5.1 Kuvaus 19
5.2 Työn toteutus 20
5.3 Tiedot 21
5.3.1 Rakenne 21
5.3.2 Jäsenmuuttujat 22
5.4 Toiminnot 25
5.5 Muut ominaisuudet 27
5.5.1 Yleiset rajoitteet 27
5.5.2 Suorituskyky 28
5.5.3 Ylläpidettävyys 28
5.5.4 Jäsenmuuttujien ja viestien välinen yhteys 29
5.5.5 Virhetilanteet 30
6 TESTAUS 31
7 YHTEENVETO 32
LÄHTEET 33

TERMIT JA LYHENTEET

Perl	Perl (*Practical Extraction and Report Language*) on työssä käytettävä ohjelmointikieli. Perl on tulkattava proseduraalinen ohjelmointikieli.
XML	XML (*eXtensible Markup Language*) on kuvauskieli, tai tapa kuvata jotain tietoa.
HTML	HTML (Hypertext Markup Language) on kuvauskieli, jolla voidaan kuvata hyperlinkkejä. Käytetään esimerkiksi WWW-sivujen teossa.
CSS2	CSS (Cascading Style Sheets) on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji. CSS:llä pyritään kuvaamaan www-sivujen ulkoasua.
Javascript	Web-ympäristössä käytettävä ohjelmointikieli.
WWW	WWW (World Wide Web) on Internetissä toimiva hypertekstijärjestelmä.
HTTP POST	HTTP POST on hypertekstisiirtojärjestelmän metodi. Post-metodi on lähetystapa datan lähetykseen kohdepalvelimelle.
JSP	JavaServer Pages (JSP) on Sun Microsystemsin kehittämä Javaan perustuva menetelmä luoda HTML- ja XML muotoisia websivuja.
Ajax	Ajax (akronyymi sanoista Asynchronous JavaScript And XML) on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.
MYSQL	MySql on relaatiotietokantaohjelmisto. MySql on saavuttanut eniten suosiota mm. webpohjaisissa sovelluksissa.
Apache Tomcat	Apache on avoimen lähdekoodin WWW-palvelinohjelmisto. Tomcat lisäosa on Apachen laajennus, joka tarjoaa tuen Java-pohjaisten sivujen näyttämiseen muiden lisäksi.

ASCII	Ascii (akronyymi sanoista American Standard Code for Information Interchange) on 7-bittinen eli 128 merkkipaikan laajuinen tietokoneiden merkistö, joka sisältää ensisijaisesti amerikanenglannissa tarvittavat kirjaimet, numerot, väli- ja erikoismerkkejä sekä eräitä ohjauskoodeja.
Silppu	Hakurakenne eli assosiaatiotaulu (engl. associative array, map tai dictionary) on abstrakti tietotyyppi, joka kuvaa avaimia arvoiksi. Kun hakurakenteelle antaa avaimen (esimerkiksi henkilön nimi), se kertoo arvon (puhelinnumero). Perl-kielessä tunnetaan myös nimellä hash.
MD5	MD5 on niin kutsuttu message-digest-algoritmi, jota käytetään muun muassa kryptografiassa.
Kutsuva ohjelma	Ohjelma, johon on luotu ilmentymä tiedonkeruukirjastosta.

1 JOHDANTO

Tämän opinnäytetyön aiheena on suunnitella ja luoda tiedonkeruukirjasto Enfo Oyj:n tuotannonseurantajärjestelmään. Enfo on pohjoismainen IT-palvelutalo, joka tarjoaa yrityksille ja yhteisöille mutkattomia tietotekniikkapalveluja. Enfo hyödyntää palveluissaan yli 45 vuoden kokemustaan tietotekniikasta sekä vahvojen IT-ammattilaistensa osaamista. Lähes 750 huippuosaajaa varmistavat, että asiakkaat saavat parhaimman hyödyn irti tietotekniikasta. Enfon liikevaihto vuonna 2011 oli noin 140 milj. euroa. Yrityksellä on toimipisteitä Suomessa Kuopiossa, Espoossa, Jyväskylässä, Lahdessa ja Tampereella. Enfo tarjoaa seuraavia palveluja: IT-Infrastruktuurin hallintapalvelut, sovellusten hallintapalvelut, tiedonvälityspalvelut, konsultointi ja projektiratkaisut, laitteet ja lisenssit, järjestelmäintegraatiopalvelut ja toiminnanohjausjärjestelmäpalvelut.

Työ tehtiin Enfo Oyj:n Tiedonvälityspalvelut-yksikössä osana projektiryhmää, jonka tehtävänä oli arvioida uuden tuotannonseurantajärjestelmän toimivuutta yrityksen tuotannonseurantaprosessissa. Projektin tavoitteena oli implementoida toimiva kokonaisuus tuotannonseurannasta sekä myyntilaskujen toimitusprosessista.

Työn tarkoituksena on suunnitella ja toteuttaa Enfon järjestelmiin liitettävä tiedonkeruukirjasto, joka kerää tietoa eri prosesseista ja lähettää ne tuotannonseurantajärjestelmälle. Kohdejärjestelmät, joihin kirjastot liitettäisiin, ovat erilaisia tuotantoprosessissa olevia ohjelmia. Vaatimuksina on, että kirjaston tulee olla helppokäyttöinen ja että sen pystyisi liittämään helposti muihin järjestelmiin.

Opinnäytetyön raportti jakaantuu kolmeen osaan: yleisiin asioihin ja tekniikoihin, tehtyyn työhön sekä lopputestaukseen ja yhteenvetoon. Ensimmäisessä osassa esitellään projektia, jonka tarkoituksena oli suunnitella ja toteuttaa toimiva kokonaisuus tuotannonseurantajärjestelmästä. Esiteltäviä asioita yleisesti ovat projektin kuvaus, henkilöstön roolien kuvaus sekä suunniteltu aikataulu. Osiossa kuvataan myös tuotannonseurantajärjestelmän tarkoitusta, toimintaa ja sinne lähetettäviä viestejä. Toiminnassa esitellään, kuinka viestit luetaan ja mitä niille tapahtuu järjestelmän sisällä.

Toisessa osiossa esitellään työmenetelmissä käytetyt vaiheet kuten suunnittelu, määrittely ja toteutus. Osiossa kuvataan tarkemmin tiedonkeruukirjaston toiminnallisuutta sekä sen sisältämiä tietotyyppejä. Lisäksi tarkemmin kuvataan tiedonkeruukirjaston käyttöominaisuuksia sekä sen sopivuutta kohdejärjestelmän kanssa.

Viimeisessä osassa on keskitytty tiedonkeruukirjaston testaukseen. Osiossa esitellään luotujen sanomien oikeellisuutta, muihin järjestelmiin liitettävyyttä sekä virheidenkäsittelyä. Loppuyhteenvedossa analysoidaan työn lopputulosta ja tavoitteita sekä sitä, onko lopputulos odotettu ja voiko kirjastoa käyttää tuotantokäytössä.

2 PROJEKTI

2.1 Yleiskuvaus

Projektin ideana oli kehittää ympäristö, jolla arvioitaisiin uuden tuotannonseurantajärjestelmän käytettävyyttä yrityksen tiedonvälityspalveluissa tuotannonseurantaprosessissa. Tuotannonseurantaprosessi käsittää aineistojen käsittelyn aina raakadatasta esimerkiksi postitukseen tai sähköpostitukseen asti. Aineistot voivat olla esimerkiksi asiakkaiden lähettämiä laskutusaineistoja, jotka sitten postitetaan eteenpäin. Projektin tavoitteena oli implementoida toimiva kokonaisuus tuotannonseurannasta sekä myyntilaskujen toimitusprosessista.

2.2 Henkilöstö ja roolit

Henkilöstöä projektissa oli kymmenkunta. Projektiin kuului projektipäällikkö, kaksi tiiminvetäjäjää ja kuusi tiimiläistä. Projektipäällikön tehtävänä oli ohjata tiimiä ja raportoida johtoryhmälle. Tiiminvetäjien tehtävänä oli valmentaa projektijäseniä sekä suunnitella ja toteuttaa kehitysratkaisua niin teknisesti kuin yleisesti. Muiden jäsenten vastuualueina olivat määrätyt osa-alueet, joiden perusteella tuotannonseurantaympäristöä kehitettäisiin. Työtehtäviin kuului lisäksi dokumentointi.

Oma osani projektista oli suunnitella ja toteuttaa järjestelmiin liitettävä tiedonkeruukirjasto, joka kerää tietoa eri prosesseista ja lähettää ne tuotannonseurantajärjestelmälle. Kohdejärjestelmät, joihin kirjastot liitettäisiin, olivat erilaisia tuotantoprosessissa olevia ohjelmia. Vaatimuksina oli, että kirjaston tulee olla helppokäyttöinen ja että sen pystyisi liittämään helposti muihin järjestelmiin. Työ sisälsi suunnittelun, määrittelyn, toteutuksen ja testauksen. Lisäksi oli ongelmanselvittelyä sekä laajalti selvitystyötä.

2.3 Aikataulut

Projektin aikataulutus oli jaoteltu erilaisiin työvaiheisiin. Työvaiheiden edistystä ja niiden perusteella jatkosuunnittelua pystyttiin valvomaan workshopeilla ja palavereilla. Workshopeissa tuotannonseurantajärjestelmän toimittaneen ulkopuolisen yrityksen edustaja kävi konsultoimassa ja antamassa tietoa järjestelmän toimivuudesta ja liitettävyydestä. Workshoppien lisäksi järjestettiin projektiryhmän sisäisiä palavereja,

joiden perusteella pystyttiin arvioimaan, onko projekti aikataulussa. Sisäisiä palavereja järjestettiin viikoittain.

Tämän opinnäytetyön aikataulujen puolesta tiedonkeruukirjaston tulee olla valmiina ja testattuna viimeistään marraskuun puoliväliin mennessä. Vaatimuksiin kuuluu, että kirjaston avulla pystytään luomaan oikeanlaisia sanomia sekä kirjastolle on oltava kommentoinnit ja käyttöohjeet. Alla olevassa taulukossa kuvataan projektin suunniteltua aikataulua (taulukko 1).

TAULUKKO 1. Projektin aikataulut

Stages	**Description**	**Start End**
Pre-project	Prioritize and scope	12.8.2011
Initiation	Startup Project	15.8.2011 - 17.8.2011
Managing a Stage boundary	Plan product descriptions and work packages in details	15.8.2011 - 9.9.2011
Subsequent Delivery	Workshop and Design Concept	13.9.2011 - 14.9.2011
Subsequent Delivery	Process Consultation for the Production monitoring Platform	13.9.2011 - 14.9.2011
Subsequent Delivery	Deliver products	19.9.2011 - 14.10.2011
Subsequent Delivery	Installation and Configuration (Implementation) of the Pilot Project	10.10.2011 - 14.10.2011
Subsequent Delivery	Piloting	17.10.2011 – 28.10.2011
Final Delivery	Close project	24.10.2011 – 31.10.2011
	Proof Of Concept	1.11.2011 – 28.2.2012

3 KÄYTETYT TEKNIIKAT

3.1 Perl

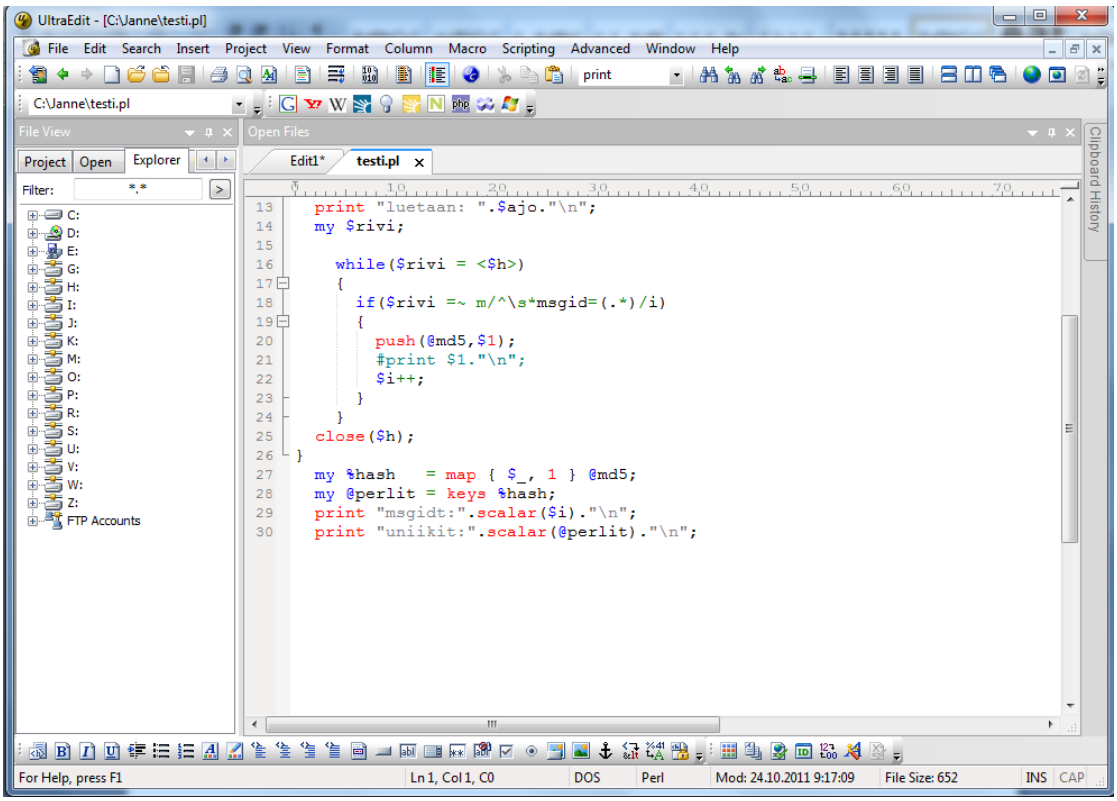
Perl-ohjelmointikieli on tarkoitettu niille, jotka tahtovat saada valmista aikaan. Ensimmäisen version suunnitteli tunnettu tietokonemies Larry Wall. Alun perin Perlin tarkoitus oli yhdistää erilaisia työvälineohjelmistoja niin, että ne yhdessä muodostaisivat kokonaisuuden, joka on enemmän kuin osiensa summa. Vähitellen Perlistä kehittyi korvaamaton apuväline esimerkiksi Windows NT -ympäristöön järjestelmähoitajien ja ohjelmoijien käyttöön. Nyt Perlin hyödyllisyys on tunnustettu tosiasia. (Randal, Olson & Christiansen 1997, 1.)

Perl on yleiskäyttöinen ohjelmointikieli, joka alun perin luotiin tekstin manipulointiin, mutta on nyt käytössä järjestelmävalvojilla, web-ohjelmien tuotannossa, tietoverkko-ohjelmoinnissa, käyttöliittymätuotannossa ja monissa muissa asioissa.

Kielen on tarkoitettu olevan käytännöllinen ja tehokas. Sen tärkeimmät ominaisuudet ovat, että se on helppo käyttää, tukee sekä proseduraalista että olio-ohjelmointia, sisältää tehokkaan sisäänrakennetun tuen tekstinkäsittelylle ja yhden maailman mahtavimmista kokoelmista kolmansien osapuolien moduuleita.

3.2 Ultraedit

Työtä aloitettaessa päätettiin, että kirjaston kehitykseen käytettäisiin Ultraedit-sovelluskehityseditoria. Ultraedit on IDM Computer Solutionsin kehittämä kaupallinen tekstieditori, joka on suunnattu laajaan ohjelmointikäyttöön. Ohjelmaa alettiin kehittää vuonna 1994 ja nykyisin siitä on julkaistu 17 versiojulkaisua. Ultraeditistä on julkaistu versiot Windows-, Linux- ja Mac-käyttöjärjestelmille. Alla olevassa kuvassa esitellään esimerkkiohjelman tekoa UltraEditissä (kuva 1).



KUVA 1. UltraEditin editorinäkymä

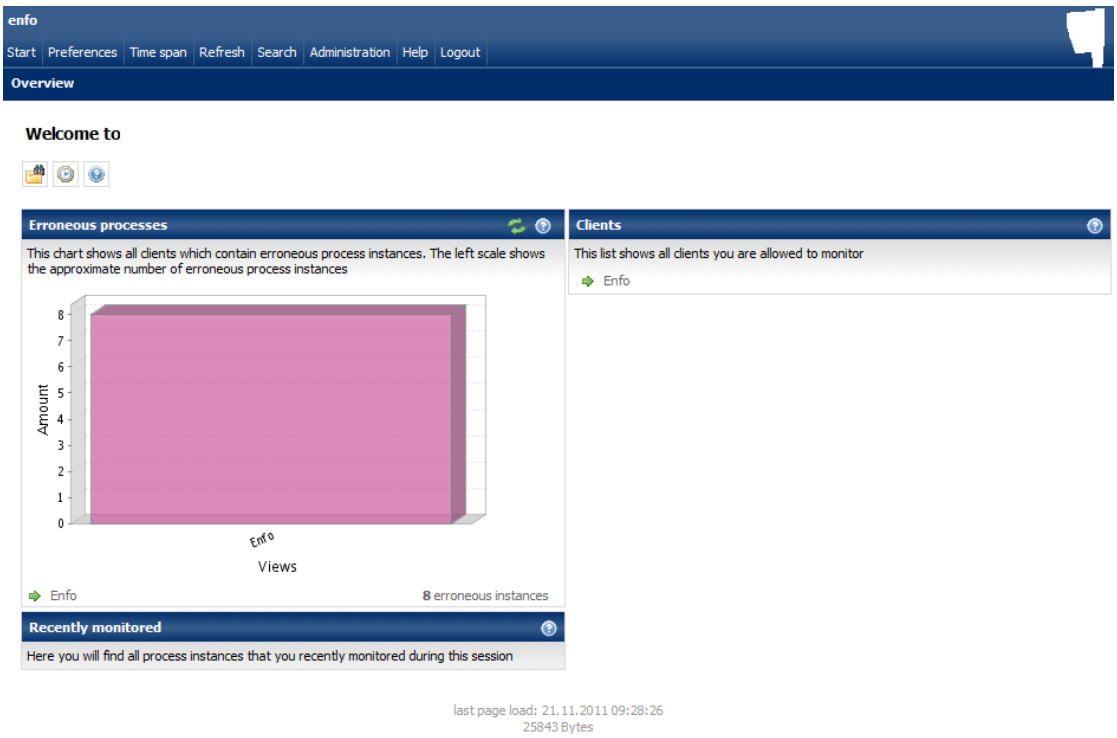
4 TUOTANNONSEURANTAJÄRJESTELMÄ

4.1 Kuvaus

Tuotannonseurantajärjestelmät on tarkoitettu esimerkiksi erilaisten prosessien tilojen seuraamiseen. Tuotannonseurannalla pyritään saamaan mahdollisimman tarkka kuvaus prosessien eri vaiheista. Tarkka tuotannonseuranta on tärkeää, koska siitä saatava tieto on tärkeää esimerkiksi johtoportaalle, ylläpidolle, asiakaspalveluun tai laskutukseen. Järjestelmien avulla pystytään seuraamaan yksittäisten tai monien prosessien tilaa, selaamaan tilastoja ja selaamaan suorituskykyä. Seurannan tuottamien tietojen pohjalta voidaan myös havaita mahdollisten ongelmatilanteet.

Tässä projektissa käytettävä tuotannonseurantajärjestelmä perustuu yrityksen tiedonvälityspalveluiden aineistojen käsittelyn seuraamiseen. Aineistot voivat olla kirjeitä, b2b-laskuja, e-laskuja, sähköpostilaskuja, netposteja tai muuta sanomaa tai liikennettä. Aineistojen käsittely tarkoittaa käytännössä aineiston käsittelyn niin sähköisesti kuin paperiaineistona. Käsittely tarkoittaa aineiston käsittelyä asiakkaan lähettämästä raakadatasta aina tulostukseen ja postitukseen asti tai sitten toimittamisen sähköisenä eteenpäin.

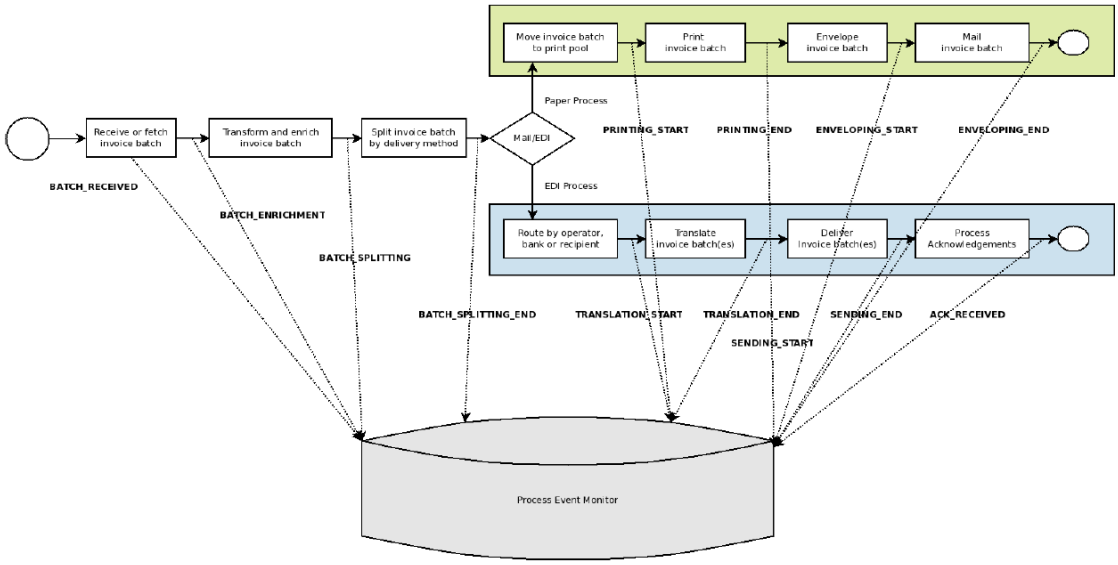
Järjestelmässä on käytetty lukuisia tekniikoita. Järjestelmä pohjautuu täysin Java-ohjelmointipohjalle, jonka päälle on rakennettu WWW-pohjainen käyttöliittymä, joka on toteutettu käyttäen XML, HTML, CSS2, JSP, Javascript ja Ajax -tekniikoita. Järjestelmän tietokantana on MySQL ja käyttöjärjestelmänä Linux. WWW-palvelimena on Apache Tomcat. Alla olevassa kuvassa on esimerkkinäkymä järjestelmän käyttöliittymästä (kuva 2).



KUVA 2. Järjestelmän etusivunäkymä

4.2 Toiminta

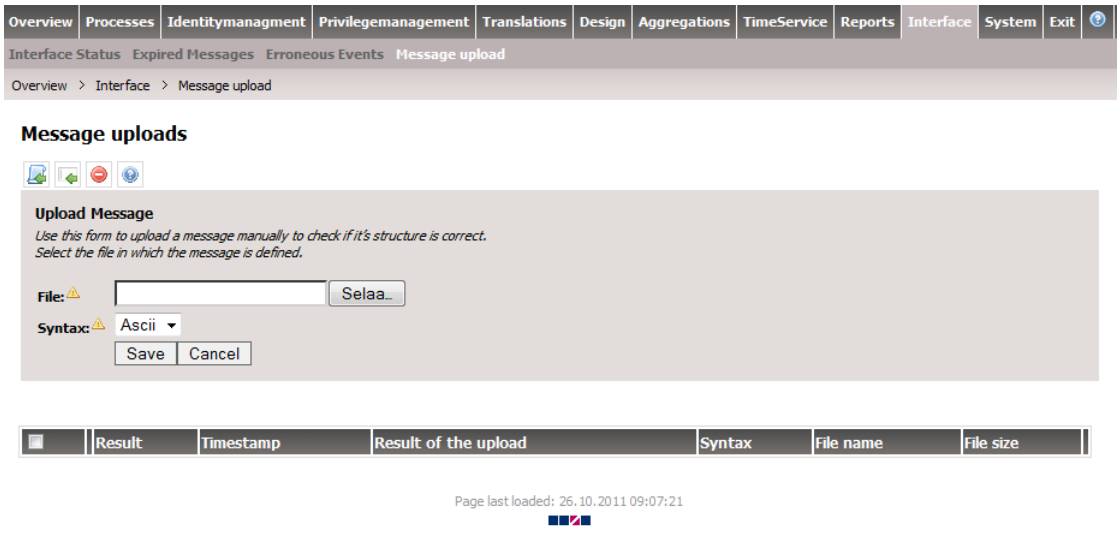
Tuotannonseurantajärjestelmän toiminta perustuu siihen lähetettyihin sanomiin. Järjestelmässä on vastaanottotoiminto, johon käsiteltävät sanomat voidaan lähettää ja josta ne käsitellään. Sanomat ovat XML:ää muistuttavia arvopareja, jotka kuvaavat jonkin prosessin vaihetta. Järjestelmälle lähetettäviä sanomia voidaan toimittaa koko prosessin ajan, ja järjestelmä osaa linkittää ne yhteen attribuuttien yksilöllisten tunnistearvojen kautta. Sanomat generoidaan lähettävän prosessin tietojen perusteella. Kun järjestelmä on vastaanottanut sanoman, se laittaa sen käsittelyjonoon, josta se käsitellään viestityypin mukaan. Viestit lajitellaan uniikkien tunnisteiden mukaan eräänlaiseen prosessipuuhun, jossa on aikuis- ja lapsielementtejä. Alla olevassa kuvassa esitellään esimerkkikaavio lähetettävistä sanomista (kuva 3).



KUVA 3. Järjestelmälle lähetettäviä sanomia prosessista

4.3 Sanomat

Tuotannonseurantajärjestelmä käyttää käyttöliittymässä ASCII-pohjaista formaattia sanomien liikuttelussa. ASCII-pohjainen formaatti käyttää tietojen syötössä arvopare-ja, joilla on nimi ja arvo. Kun sanoma lähetetään järjestelmälle, niin järjestelmä tekee tarvittavat toiminnot sanomien attribuuttien mukaan. Sanomia voidaan lisätä järjes-telmään joko käsin käyttöliittymän kautta lataamalla tiedosto kiintolevyltä, tekstikentän kautta tai ohjelmallisesti suoraan POST-tyyppisellä HTTP-pyynnöllä. Allaolevassa kuvassa on käsin lähetettävän sanomatiedoston käyttöliittymänäkymä (kuva 4).



KUVA 4. Sanoman lähetysnäkymä manuaaliselle lähetykselle

Järjestelmässä on viittä eri sanomatyyppiä: addrecord, addtask, context, modifrecord ja modifcontext. Addrecord luo järjestelmään uuden prosessi-ilmentymän annetulla recordtypellä. Addtask luo uuden tapahtuman jo ennestään luotuun prosessi-ilmentymään. Context-tyypin viesti ohjeistaa järjestelmää luomaan relaation kahden jo luotujen prosessi-ilmentymien välille. Modifrecord muokkaa jo järjestelmään tehtyä prosessi-ilmentymää. Modifcontext muokkaa jo järjestelmään tehtyä contextia tai toiselta nimeltään relaatiota. Alla olevassa kuvassa esitellään addrecord-tyypin esimerkkisanoma (kuva 5).

```
01 *** ADDRECORD - BEGIN ***
02 MonitoringMsgType=ADDRECORD
03 TypeKey=INVOICE
04 BPId=TXP
05 Value=122155208427704
06 Attr_CC4ID=CC4_122155208427704
07 Attr_CC4_RECIPIENT=11673977
08 Attr_COSTCENTER=33543
09 Attr_CHARSET_SRC=B
10 Attr_CHARSET_TARGET=A
11 Attr_RECEIPIENT=COMB010
12 Author=CC4
13 BPTS=2008-09-16 10:01:24
14 MsgId=RECORD#0#12215520842770#00004
15 *** ADDRECORD - END ***
```

KUVA 5. Esimerkki addrecord-tyypin sanomasta

Sanoman runko koostuu seuraavista lohkoista:

- Monitorin viestityyppi
- Header
    - tyyppiavain joka on prosessityyppi tai tapahtumatyyppi
    - avainattribuutit, joita on nolla tai enemmän
    - kontekstiavain, vain jos sanomiin liittyy relaatioita
- Payload
    - tyyppiavain joka on prosessityyppi tai tapahtumatyyppi
    - avainattribuutit, joita on nolla tai enemmän
    - attribuutit

5 TIEDONKERUUKIRJASTO

5.1 Kuvaus

Tiedonkeruukirjasto on Perl-kielellä tuotettu luokka tai toiselta nimeltä moduuli. Sen toiminta perustuu siihen, että siitä tehdään ilmentymä jossain muussa ohjelmassa ja sille annetaan komentoja sekä syötetään arvoja erilaisten funktioiden avulla. Kirjasto on tarkoitettu tuotantokäytössä olevien prosessien datan seurantaan ja lähetykseen tuotannonseurantajärjestelmälle. Tarkoituksena on liittää se prosessia eteenpäin ajaviin ohjelmiin, josta se keräisi seurattavan datan ja muokkaisi sen oikeaan muotoon, jotta tuotannonseurantajärjestelmä pystyisi lukemaan sitä. Kirjaston luentaan liittyy tietyt pakolliset tiedot sekä vapaaehtoisesti annettavat ja nimettävät attribuutit.

Kirjaston perustoimintoihin kuuluu muuttujien alustus, monitoroinnin aloitus, tietojen syöttö, tietojen tulostus, monitoroinnin lopetus ja sanomien lähetys. Kun aletaan luoda oliota luokkakirjastosta, alustetaan muuttujat, jolloin esimerkiksi olio saa yksilöllisen prosessitunnistemuuttujan. Monitoroinnin aloituksessa luodaan automaattisesti uusi tiedosto ja asetetaan monitorointitila aktiiviseksi. Tiedoston luonti tapahtuu yksilöllisesti aikaleiman ja prosessi tunnistemuuttujan mukaan. Ennen monitoroinnin alustusta täytyy olla oliolle syötetty seuraavat kansiot: output-, väliaikais- ja varmuuskopiokansio. Kun monitorointi on aktiivinen, on tietojen syöttö oliolle mahdollista. Tietojen tulostuksessa laitetaan aina yksittäinen tai useampi sanoma jo ennestään avattuun tiedostoon. Tietojenseurantajärjestelmä vaatii tietyt vaaditut attribuutit olevan olemassa, joten jonkin tiedon puuttuessa laitetaan jokin sanoma virhetilaan, jonka järjestelmä sivuuttaa. Monitoroinnin lopetuksessa suljetaan tiedosto, asetetaan monitorin tila ei-aktiiviseksi ja siirretään luotu tiedosto outputkansioon odottamaan lähetystä. Sanomien lähetyksessä on tarkoituksena siirtää luotu tiedosto outputkansiosta tuotannonseurantajärjestelmälle. Onnistuneen tiedostonsiirron jälkeen luotu tiedosto siirretään varmuuskopiokansioon. Tässä voi syntyä monia erilaisia virhetilanteita. Esimerkiksi jos tuotannonseurantajärjestelmään ei saada yhteyttä tai järjestelmän palauttama vastaus on virheellinen, laitetaan tiedosto onhold-tilaan. Muita virheitä voivat olla esimerkiksi tiedostojärjestelmän ongelmat, määrittelemättömät kansiot, luokan virheellinen käyttö tai jonkin muun ohjelman vuoksi tapahtunut virhe. Kirjaston kirjoituksen voi myös abortoida erillisellä komennolla, jolla kirjoitus lopetetaan välittömästi ja luotu tiedosto hävitetään.

5.2 Työn toteutus

Työ toteutettiin vuonna 2011 syksyllä Enfolla. Työn toteutus tehtiin työpaikalla, eikä etätyömahdollisuutta ollut salassapitosyiden takia. Koska olin suorittanut jo aikaisempaa työharjoittelua yrityksessä, järjestelmät ja toimintaperiaatteet olivat tuttuja, joten pääsin suoraan tutustumaan kohdeprojektiin sekä tuotannonseurantajärjestelmään.

Työtä alettiin tehdä projektin aikataulujen mukaisesti. Työn kulkuun kuului seuraavat vaiheet: suunnittelu, määrittely, toteutus ja testaus. Suunnitteluvaihe koostui alustavista määrittelyistä, kuten viestien rungosta ja yleisestä toimintatavasta. Määrittelyosuudessa lisättiin vaatimuksia, kuten tarkempaa viestirunkoa, virhetilanteita, menettelytapoja ja käytettävyyttä. Määrittelyosuudessa päädyttiin siihen, että kirjaston täytyy olla mahdollisimman helppokäyttöinen ja tarpeeksi nopea muistinkäytön ja viestien kirjoitusnopeuden kannalta. Toteutusvaiheessa, kun kirjastoa alettiin ohjelmoida, kehitys tapahtui versioittain ja tulosta seurattiin jatkuvilla katselmoinneilla ja palavereilla. Toteutus aloitettiin järjestelmän testauksessa käytettyjen tiettyjen malliviestien pohjalta. Näitä viestejä apuna käyttäen pystyttiin luomaan runko tiedonkeruukirjaston viestiosuudelle. Testausvaiheessa kirjasto liitettiin muutamiin yrityksen järjestelmiin, joista testisanomia voitiin generoida tuotannonseurantajärjestelmälle. Testausvaiheiden perusteella, kirjastoa täytyi päivittää ja ongelmia korjata aina tarpeen tullessa. Luokkakirjaston lisäksi dokumentoitiin projektisuunnitelma, määrittelydokumentti ja testaussuunnitelma.

Kirjaston ensimmäisessä versiossa oli luotu pelkästään perusrunko viesteille sekä niiden tulostus. Testaus kuitenkin osoitti, että helppokäyttöisyys vielä puuttuu ja että kontrollia tarvitaan enemmän. Kirjaston toiseen versioon lisättiin settings-osio, jolla pystytään kontrolloimaan tarkemmin kirjaston toimivuutta. Lisäksi mukaan lisättiin lähetysfunktiot tuotannonseurantajärjestelmää varten sekä virheidenkeruu. Kolmannessa ja neljännessä versiossa kirjaston muutokset olivat lähinnä hienosäätöä lopputuotetta varten.

5.3 Tiedot

Kirjaston tiedot on tallennettu ohjelman sisässä määriteltäviin muuttujiin. Näiden tietojen toimintaperiaate on se, että yksi tietuelohko kuvaa sanomia ja toinen muuta toiminnallisuutta.

5.3.1 Rakenne

Tiedonkeruukirjaston rakenne koostuu kahdesta lohkosta: message ja settings. Message-lohko sisältää erilaisia jäsenmuuttujia, joilla kuvataan kirjoitettavan viestin runkoa. Settings-lohko koostuu yleisistä jäsenmuuttujista, joita käytetään kirjaston sisäisen toiminnallisuuden määrittämiseen. Koska Perl-kielessä ei olio-ohjelmointi ole suurelta osin tuettu, on jäsenmuuttujat suunniteltu säilöttäväksi silppu-tyyppiseen elementtiin, joka sisältää kaksi kappaletta sisäkkäisiä silppuja (kuva 6).

```
my $self =
{
  message =>
  {
    _Mandant => undef, #string
    _HeaderMonitoringMsgType => undef, #string
    _PayloadMonitoringMsgType => undef, #string
    _HeaderTypeKey => undef, #string
    _PayloadTypeKey => undef, #string
    _ContextKey => undef, #string
    _HeaderAttributes => undef, #hash, string keypairs
    _BPId => undef, #string
    _Value => undef, #string
    _PayloadAttributes => undef, #hash, string keypairs
    _Author => undef, #string
    _BPTS => undef, #string
    _MsgId => undef, #string
    _Description => undef, #string
  },
  settings =>
  {
    _tmpfolder => undef, #string
    _outputfolder => undef, #string
    _backupfolder => undef, #string
    _filename => undef, #string
    _filehandle => undef, #file handle variable
    _messagecounter => undef, #int
    _processid => undef, #string
    _state => undef, #0 <- stopped, 1 <- started
    _errors => undef, #array
  },
};
```

KUVA 6. Kirjaston tietojen rakenne

5.3.2 Jäsenmuuttujat

Jäsenmuuttujat on jaoteltu kahteen taulukkoon. Jäsenmuuttujia käytetään ja arvoja syötetään tai määritetään joissain tapauksissa automaattisesti ja joissain tapauksissa käyttäjä antaa arvon. Jäsenmuuttujien pakollisuus voi riippua erilaisista tapauksista. Alla olevista taulukoista selviää muuttujien käyttötapaukset (taulukko 2, taulukko 3).

Luonti- ja pakollisuussarakkeen tarkastus message-osiossa suoritetaan tulostuskomentoa kutsuessa. Settings-osiossa luonti- ja pakollisuustarkastukset suoritetaan automaattisesti aina monitorin tilan mukaisesti.

TAULUKKO 2. Message-muuttujat

Nimi	**Kuvaus**	**Tyyppi**	**Luonti**	**Pakollisuus**
Mandant	Mandant-arvo	Merkkijono	Kutsuva ohjelma tai automaattisesti arvon ollessa tyhjä	X
HeaderMonitoringMsgType	Headerlohkon MonitoringMsgType	Merkkijono	Kutsuva ohjelma	X
PayloadMonitoringMsgType	Payloadlohkon MonitoringMsgType	Merkkijono	Kutsuva ohjelma	
HeaderTypeKey	Headerlohkon TypeKey	Merkkijono	Kutsuva ohjelma	X
PayloadTypeKey	Payloadlohkon TypeKey	Merkkijono	Kutsuva ohjelma	
ContextKey	Kontekstiavain	Merkkijono	Kutsuva ohjelma	
HeaderAttributes	Headerlohkon attribuutit	Silppu, Merkkijono arvoparit	Kutsuva ohjelma	
BPId	BPId-arvo	Merkkijono	Kutsuva ohjelma	X

Value	Value-arvo	Merkkijono	Kutsuva ohjelma	X
PayloadAttributes	Payloadlohkon attribuutit	Silppu, Merkkijono arvoparit	Kutsuva ohjelma	
Author	Author-arvo	Merkkijono	Kutsuva ohjelma	X
BPTS	BPTS-arvo	Merkkijono	Kutsuva ohjelma tai automaattisesti arvon ollessa väärää formaattia tai tyhjä	X
MsgId	MsgId-arvo	Merkkijono	Kutsuva ohjelma tai automaattisesti arvon ollessa tyhjä	X
Description	Description-arvo	Merkkijono	Kutsuva ohjelma	X

TAULUKKO 3. Settings-muuttujat

Nimi	**Kuvaus**	**Tyyppi**	**Luonti**	**Pakollisuus**
tmpfolder	Väliaikaiskansio johon keskeneräistä tiedosta kirjoitetaan	Merkkijono	Kutsuva ohjelma	X
outputfolder	Output-kansio johon valmis viestitiedosto siirretään tmpkansiosta	Merkkijono	Kutsuva ohjelma	X
backupfolder	Backup-kansio, johon tiedosto siirretään postmessagen jälkeen.	Merkkijono	Kutsuva ohjelma	X
filename	Tiedoston nimi	Merkkijono	Automaattisesti	X
filehandle	Tiedostokahva	Tiedostokahvamuuttuja	Automaattisesti	X
messagecounter	Viestilaskuri	Integer	Kasvatetaan onnistuneiden viestien tulostuksessa. virhetapauksissa ei kasvateta	X
processid	Skriptin prosessin id	Merkkijono	Automaattisesti	X
state	Monitorin tila	Integer	Automaattisesti	X
errors	Lista virheistä	Taulukko	Automaattisesti	

5.4 Toiminnot

Toiminnot on suunniteltu käytettävyyden kannalta mahdollisimman yksinkertaisesti. Kirjaston toimintaa ohjaavat funktiot on pyritty pitämään minimissään.

Aloitettaessa tiedonkeruuta tulee monitorin paketti tuoda kutsuvaan ohjelmaan komennolla:

```
use ENFO::Monitor;
```

sekä luoda siitä instanssi komennolla:

```
my $monitorobject = new ENFO::Monitor();
```

Kun monitorille on luotu oliomuuttuja, tulee monitorille määrittää kansiot kirjoitusta, siirtoa ja varmuuskopiointia varten. Komennot näihin ovat:

```
$monitorobject->setoutputfolder("c:\\monitor\\output\\");
$monitorobject->settmpfolder("c:\\monitor\\temp\\");
$monitorobject->setbackupfolder("c:\\monitor\\backup\\");
```

Kun kansiot on määritelty, voi monitorin käynnistää.

```
$monitorobject->Start();
```

Monitorin käynnistyksessä luodaan kirjoitettava tiedosto väliaikaiskansioon sekä määritellään tiedostonimi prosessin id:n ja aikaleiman mukaan ja laitetaan monitorin status aktiiviseksi.

Käynnistyksen jälkeen, voidaan monitorille alkaa syöttää sanomia. Jokainen sanoma generoidaan yksitellen. Tietoja sanomien eri muuttujille annetaan esimerkiksi komennoilla:

```
$monitorobject->setHeaderMonitoringMsgType("ADDRECORD");
$monitorobject->setHeaderTypeKey("INVOICE_BATCH");
```

Sanoman attribuuttien syöttö silppu-tyyppisiin muuttujiin tapahtuu komennoilla:

```
$monitorobject->setHeaderAttribute("RECEIVER_OPERATOR","OP");
$monitorobject->setHeaderAttribute("TYPE","Original");
$monitorobject->setPayloadAttribute("TEST","VALUE");
```

Yksittäisen sanoman jälkeen, tulee sanoma lähettää tiedostoon kirjoitettavaksi komennolla:

```
$monitorobject->Print();
```

Ennen kuin seuraavan sanoman arvojen syöttö alkaa, on suositeltavaa Print()- funktion jälkeen tyhjentää edellinen message-tietue komennolla:

```
$monitorobject->Clear();
```

Clear()- funktio tyhjentää vain message-osion tiedot, mutta ei settings-osiota.

Kun kirjoitus halutaan lopettaa ja siirtää tiedosto lähetyskansioon, kutsutaan komentoa:

```
$monitorobject->Stop();
```

Stop()- funktio asettaa monitorin tilan myös ei-aktiiviseksi.

Kirjoitus voidaan myös abortoida, jolloin kirjoitetut viestit ja luotu tiedosto tuhotaan täysin ja monitorin tila laitetaan ei-aktiiviseksi komennolla:

```
$monitorobject->Abort();
```

Monitoroinnin lopetuksen jälkeen voidaan luotu tiedosto siirtää tuotannonseurantajärjestelmälle komennolla:

```
$monitorobject->Send();
```

Send()- funktio lähettää tiedoston järjestelmälle, ja onnistuneiden siirtojen jälkeen siirtää tiedostot varmuuskopiokansioon. Jos viestin lähetys epäonnistuu, laitetaan tiedosto odotustilaan nimeämällä se _onhold-päätteellä. Send()- vaihe tarkastaa myös mahdolliset odottavat tiedostot varsinaisen sanomatiedoston lisäksi.
Tiedoston lähetyksen jälkeen voi tarvittaessa tarkastaa, onko syntynyt mahdollisia virheitä ylläkuvattuna olevan prosessin aikana. Virheet palautetaan taulukkotyyppisenä merkkijonoina komennolla:

my @errors = $monitorobject->geterrors();

Kirjasto sisältää siis yksinkertaisuudessaan vain pienen määrän komentoja, joista käyttäjän tarvitsee huolehtia. Kirjaston toiminta on muilta osin sisäisesti automatisoitu, joiden toiminta ei näy käyttäjälle.

5.5 Muut ominaisuudet

5.5.1 Yleiset rajoitteet

Kirjaston oikeanlaiseen toimintaan on määrätty monenlaisia rajoitteita. Kohdejärjestelmän täytyy olla Windows 98, tai uudempi, joka tukee win32 moduuleita tai Linux/Unix. Kohdejärjestelmässä täytyy olla internetyhteys. Kohdejärjestelmän tulee sijaita Enfo Oyj:n sisäverkossa sekä sillä tulee olla pääsy tuotannonseurantapalvelimelle. Kohdejärjestelmässä tulee olla asennettuna Perl sekä seuraavat kolmannen osapuolen moduulit: Time-HiRes, File-Copy, LWP-UserAgent, Digest-MD5 ja HTTP-Request-Common. Kirjastomoduulin tulee olla asennettuna ENFO-kansioon. Kutsuvan ohjelman tulee olla samassa juuressa, jossa ENFO-kansio sijaitsee. Järjestelmässä täytyy olla olemassa kansiot väliaikais-, siirto- ja varmuuskopiointikansioille niitä määritettäessä.

5.5.2 Suorituskyky

Kirjasto on suunniteltu mahdollisimman suorituskykyiseksi nopeuden ja muistinkäytön kannalta. Suunnittelussa tehokkaaseen muistinkäyttöön päädyttiin sillä periaatteella, että muistia tarvitsee varata mahdollisimman vähän ja että tämän toimintaperiaatteen sijasta pyritään tyhjentämään ja kierrättämään jo ennestään käytössä olevia muuttujia. Tulostustoiminnan suorituskykyä voitiin parantaa tarkastamalla virheviestit ja jättämällä ne tulostamatta kokonaan, jolloin voitaisiin siirtyä seuraavaan sanomaan nopeammin. Kirjasto on kykenevä tulostamaan samanaikaisesti useita satojatuhansia sanomia sekä luomaan jokaiselle sanomalle yksilöllisen tunnisteen. Yksilöllinen tunniste on kaksinkertainen MD5-hash, jonka generointi tapahtuu mm. aikaleiman ja prosessi-idn mukaan.

5.5.3 Ylläpidettävyys

Ylläpidettävyys oli yksi tärkeimmistä pääasioista, kun kirjaston toimintaa alettiin suunnitella. Jos kirjastoon joudutaan lisäämään toiminnallisuutta tai poistamaan vanhoja tai muokkaamaan nykyisiä, niiden muutosten tekeminen tuli olla nopeata ja vaivatonta. Tämän johdosta toteutus on tehty mahdollisimman selkeillä olio-ohjelmoinnin perusperiaatteilla. Jokaiselle toiminnolla on oma funktionsa sekä jäsenmuuttujalle asetus- ja hakufunktionsa. Kirjaston jokainen vaihe on kommentoitu ja siitä on tehty määrittelydokumentti ja käyttöohjeistus. Esimerkiksi testausvaiheessa ja versiopäivityksissä ylläpidettävyys oli tärkeässä asemassa.

5.5.4 Jäsenmuuttujien ja viestien välinen yhteys

Alla olevassa taulukossa verrataan järjestelmälle lähetettävien sanomien arvoja ja niitä vastaavia jäsenmuuttujia (taulukko 4).

TAULUKKO 4. Esimerkki CONTEXT-tyypin sanomasta

Viestiosa	**Jäsenmuuttuja**
*** CONTEXT - BEGIN ***	aloitusotsikko, käyttää header-monitoringmsgtypeä
MANDANT=UI	Mandant
MonitoringMsgType=CONTEXT	HeaderMonitoringMsgType
TypeKey=BATCH	HeaderTypeKey
ContextKey=CONTAINS	ContextKey
BATCH_ID=Sample text value 1 (KM)	HeaderAttribute(s)
MonitoringMsgType=RECORD	PayloadMonitoringMsgType
TypeKey=INVOICE	PayloadTypeKey
INVOICEID=Sample text value 2 (KM)	PayloadAttribute(s)
PAYEE_NETSERVICE_ID=Sample text value 3 (KM)	PayloadAttribute(s)
BATCH_ID=Sample text value 4 (M)	PayloadAttribute(s)
INVOICE_DATE=20110916	PayloadAttribute(s)
BPId=BATCH_CONTAINS_INVOICE_SAMPLE (MS)	BPId
BPTS=20110916 105201 (MS)	BpTS
Value=an identifying value for process 'INVOICE' (MS)	Value
Author=enfo (MS)	Author
MsgId=BATCH_CONTAINS_INVOICE_SAMPLE#1 (S)	Msgid
Description=Here you can send a description or comment. (S)	Description
*** CONTEXT - END***	lopetusotsikko, käyttää header-monitoringmsgtypeä

5.5.5 Virhetilanteet

Kirjastoon toteutetut virheentarkistukset on pyritty tekemään mahdollisimman tarkasti mutta yksinkertaisesti. Mahdolliset virhetilanteet on karsittu minimiin, joten virheen sattuessa se on helppoa jäljittää. Alla olevassa taulukossa on virhetilanteet selitettynä (taulukko 5).

TAULUKKO 5. Virhenumeroinnit

Numero	**Virhekuvaus**	**Funktio**
1	Outputtiedoston luonti epäonnistui	Start()
2	Output-, tmp tai backupkansiota ei ole määritelty	Start()
3	Monitori on jo startattu	Start()
4	Monitoria ei ole startattu	Print()
5	Viallinen serverivastaus tuotannonseurantajärjestelmästä. Tapahtuu tiedoston lähetyksessä. Virhe muodostuu järjestelmän palauttaman XML tiedoston perusteella. Jos esim filesize on null tai filename on unkown tai jos järjestelmän palauttama XML ei ole message-monitorigresponse.	Send()
6	Serveriin ei saada yhteyttä	Send()
7	Onhold-lähetyksessä yritettiin lähettää tiedostoa, jota ei ole olemassa tai kyseisessä tiedostossa ei ole _onhold päätettä	Send()

6 TESTAUS

Testauksen tavoitteena oli arvioida kirjaston käytettävyyttä, toiminnallisuutta ja luotettavuutta tuotantokäytössä sekä tarkastella yhteensopivuutta erilaisten järjestelmien kanssa. Lisäksi testaukseen kuului paikallista toiminnallisuustestausta ja järjestelmätestausta suorituskyvyn kannalta.

Kirjaston toiminnallisuutta luotaessa jatkuva testaus oli osa prosessia. Automaattisten arvojen generointi tai virheilmoitukset eivät saaneet kaataa kutsuvan ohjelman toimintaa. Ensimmäinen osa testiprosessia oli testata ja selvittää tuotannonseurantajärjestelmän sanomien toiminta. Järjestelmään syötettiin manuaalisesti yksittäisiä mallisanomia, joiden toiminnallisuutta sitten analysoitiin. Tämän jälkeen pystyttiin aloittamaan paikallinen testaus kirjaston generoimilla viesteillä. Paikalliseen testaukseen kuului massaviestien luomista samanaikaisesti sekä viestien analysointia toisen testiohjelman avulla. Testiohjelmalla pystyttiin tarkastamaan yksilöllisten viestien id-arvot. Tarkastuksella pyrittiin varmistamaan, ettei kirjasto luo useita samanlaisia arvoja. Lisäksi testaukseen kuului yhteistoiminta tuotannonseurantajärjestelmän kanssa ja tulosten analysointi. Kun paikallinen testaus oli suoritettu hyväksyttävillä tuloksilla, annettiin kirjastosta silloinen versio projektiryhmälle, joka toimi testiryhmänä.

Yleiseen projektiryhmän testaukseen kuului integrointi tuotannollisiin testisovelluksiin ja niiden toiminnan testaus yhteensopivuuden ja massaviestien generoinnin näkökulmasta. Tärkeitä tarkasteltavia osa-alueita olivat: logiikka, käytettävyys, sanomien luonnin selkeys, kirjaston toimintaprosessi, virhetilanteiden käsittely ja integraatio eri käyttöjärjestelmille ja palvelimille.

Jos kirjasto ei läpäissyt testiä, se siirtyi uudelleenkehitettäväksi, jolloin tutkittiin ja korjailtiin virheitä sekä luotiin uusia toimintoja. Virheen korjailuihin saattoi kuulua niin käytettävyysseikkoja kuin ohjelman sisäiseen toimintaan liittyviä seikkoja.

7 YHTEENVETO

Opinnäytetyössä työskenneltiin osana projektiryhmää sekä luotiin prosessi, jonka tarkoituksena oli toteuttaa prosessi tuotannonseurantajärjestelmän tiedonkeruukirjaston viemiseksi tuotantokäyttöön. Prosessi sisälsi suunnittelun, määrittelyn, toteutuksen ja testauksen. Projektiryhmässä työskentelyyn kuului tiimityöskentelyä, palavereja ja katselmointeja. Lopuksi tiedonkeruukirjasto liitettiin järjestelmiin projektin määritysten mukaisesti.

Tällä hetkellä kirjaston sijoittamista tuotantokäytössä testataan. Testattavia kohteita ovat funktioiden toiminta sekä mahdolliset myöhemmin lisättävät toiminnot.

Tulevaisuudessa kirjastoon voidaan lisätä toiminnallisuutta uusia toimintoja varten tai korjauksia nykyiseen toimintamalliin. Koska tuotannonseurantajärjestelmä ei ole vielä lopullisessa versiossa, muutoksia kirjastoon voi mahdollisesti tulla.

Opinnäytetyöprosessi sujui pääpiirteittäin ongelmitta. Joitain ongelmia ilmeni testausvaiheessa ja integrointivaiheessa, kun kirjastoa alettiin liittää muihin käyttöjärjestelmiin. Suurin ongelma oli kirjaston liittäminen OpenVMS-käyttöjärjestelmään, jossa ei ole tukea suureen osaan Perlin kolmannen osapuolen moduuleita. Lopulta päätettiin, että kirjastoa ei käytetä OpenVMS:ssä, vaan kyseistä tarkoitusta varten luodaan kokonaan erilainen ratkaisu.

Yleisesti opinnäytetyön teko sujui hyvin. Koska työtä aloittaessa työpiste ja toimintatavat olivat jo aikaisemmasta työharjoittelujaksosta tuttuja ja valmiina, työ saatiin nopeasti käyntiin. Aikaisempi kokemus Perl-kielestä ja järjestelmistä auttoi kokonaisuuden nopeassa hahmottamisessa. Työtä tehdessäni opin uusia asioita projektityöskentelystä ja monista erilaisista ohjelmistotuotannon vaiheista.

LÄHTEET

Randal, L. S., Olson, E. & Christiansen, E. 1997. *Learning Perl for Win 32*. New York: O'Reilly & Associates.

Perl Programming Documentation [verkkosivu]. [viitattu 8.11.2011]. Saatavissa: http://perldoc.perl.org/perlintro.html

UltraEdit, Wikipedia [verkkosivu]. [viitattu 9.11.2011]. Saatavissa: http://en.wikipedia.org/wiki/UltraEdit


SAVONIA