Aalto University
School of Chemical
Technology

School of Chemical Technology

Degree Program of Chemical Technology

**Markus Sintonen**

# OPC UA BASED MULTIVARIATE ANALYSIS AND DATA ACQUISITION SYSTEM FOR CHEMOMETRIC APPLICATIONS

Master's thesis for the degree of Master of Science in Technology

submitted for inspection, Porvoo, 27.03. 2015

| | |
|---|---|
| Supervisor | Professor Sirkka-Liisa Jämsä-Jounela |
| Instructors | D.Sc. (Tech) Mikko Vermasvuori |
| | M.Sc. (Tech) Lauri Haapanen |

Aalto University
School of Chemical
Technology

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
**Abstract of master's thesis**

| | |
|---|---|
| **Author** Markus Sintonen | |
| **Title of thesis** OPC UA based multivariate analysis and data acquisition system for chemometric applications | |
| **Department** Department of Biotechnology and Chemical Technology | |
| **Professorship** Process Control | **Code of professorship** KE-90 |
| **Thesis supervisor** Professor Sirkka-Liisa Jämsä-Jounela | |
| **Thesis advisor(s) / Thesis examiner(s)** D.Sc. Mikko Vermasvuori, M.Sc. Lauri Haapanen | |
| **Date** 27.03.2015 **Number of pages** 101 (+9) | **Language** English |

**Abstract**

Chemical industries utilize a variety of different types of online analyzers, for example, in areas like quality monitoring and process control applications. Large production plants typically use several analyzer devices from multiple manufacturers which are employed to measure different target quantities. As manufacturers have their own proprietary protocols for accessing analyzer information it is usually only the target property estimates that are transferred to the higher level automation systems. Other analyzer data, for example spectra, are generally not in a suitable format for further action on the higher level systems.

This thesis outlines a solution to the analyzer data acquisition problem by utilizing OPC UA standard and OPC UA analyzer devices companion specification (ADI) to create a data acquisition system for analyzer information. The created system consists of an OPC UA server, which is used as a single access point for all analyzer related information. In addition, the analyzer data is collected and archived into a SQL database, which is accessible through the OPC UA server. The data acquisition system makes it convenient for the end users to access plant analyzer information using the standardized protocols. Furthermore, the OPC UA based data acquisition system can be used to integrate analyzer data with other types of process measurements. This thesis presents an example application where this type of data integration is utilized to increase the accuracy of the target quantity estimates of an analyzer. The same system also has a wide range of other potential applications, some of which are briefly examined in this work.

The literature part of the thesis mainly focuses on different aspects of chemometrics; pre-processing and multivariate modelling of the analyzer spectra. These techniques are used in the thesis' experimental part to process data obtained from Neste Oil Oy’s Porvoo refinery. The literature part also briefly examines different protocols used for the transfer of analyzer data through the automation networks.

The experimental part the thesis consists of three main parts: The first part is a case study where the refinery measurement data and the analyzer spectra are utilized to demonstrate how product quality estimates accuracies can be improved through data integration. In the second part, the analyzer data acquisition system is developed, including the provision of a separate OPC UA ADI wrapper server for ABB online analyzers. This wrapper was created in order to obtain the data from the production unit analyzers used in the case study. In the final part, a chemometric calculation platform is designed in order to implement the data processing sequence used to process the refinery data. This platform also utilizes the newly created data acquisition system through the OPC UA protocol.

**Keywords** OPC UA, Analyzer devices, Chemometrics, Data integration

Aalto University
School of Chemical
Technology

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
**Abstract of master's thesis**

**Tekijä** Markus Sintonen

**Työn nimi** OPC UA standardiin perustuva monimuuttaja-analysointi sekä tiedonkeruujärjestelmä kemometrisiin sovelluksiin

**Laitos** Biotekniikan ja kemian tekniikan laitos

**Professuuri** Prosessien ohjaus **Professuurikoodi** KE-90

**Työn valvoja** Professori Sirkka-Liisa Jämsä-Jounela

**Työn ohjaaja(t)/Työn tarkastaja(t)** TkT Mikko Vermasvuori, DI Lauri Haapanen

**Päivämäärä** 27.03.2015 **Sivumäärä** 101 (+9) **Kieli** englanti

**Tiivistelmä**

Kemianteollisuudessa käytetään monia erityyppisiä online analysaattoreita, joita hyödynnetään esimerkiksi laadunvalvonnassa sekä prosessien ohjauksessa. Eri laitevalmistajilta olevia analysaattorilaitteita voi olla suuri määrä käytössä isoissa tuotantolaitoksissa, mitaten eri kohdesuureita. Usealla laitevalmistajalla on usein omat suljetut protokollat, joilla analysaattori-informaatiota luetaan sekä käsitellään. Tyypillisesti ainoastaan mitattu kohdesuure on helposti saatavilla tuotantolaitoksen ylemmistä informaatiojärjestelmistä. Muu analysaattori-informaatio, kuten mitatut spektrit ja diagnostiikkadata, eivät siten ole helposti saatavilla ylemmällä automaatiotasoilla.

Tämä diplomityö esittää ratkaisun analysaattoritiedonkeruuongelmaan hyödyntämällä OPC UA standardia ja OPC UA analysaattorilaitespesifikaatiota (ADI), sekä luomalla näihin pohjautuvan analysaattoritiedonkeruujärjestelmän. Järjestelmä perustuu OPC UA palvelimeen, joka toimii yhteyspisteenä kaikelle laitoksella olevalle analysaattori-informaatiolle. Lisäksi järjestelmä tukee analysaattoridatan historiakeruuta SQL-tietokantaan, josta tietoa voidaan lukea OPC UA palvelimen kautta. Loppukäyttäjien näkökulmasta järjestelmä helpottaa analysaattori-informaation hallintaa. Diplomityö esittää myös käyttöesimerkin, jossa analysaattori- sekä prosessidataa hyödynnetään rinnakkain parantamaan tiettyjen laadunvalvontasuureiden estimointitarkkuutta. Luotu tiedonkeruujärjestelmä mahdollistaa dataintegraation, jossa yhdistetään mittausdataa monista erityyppisistä lähteistä.

Työn tutkimusosa käsittää lähinnä kemometriaan liittyviä datan käsittely- ja mallinnusmenetelmiä. Näitä tekniikoita hyödynnetään kokeellisessa osassa, jossa käsitellään Neste Oil Oy:n Porvoon jalostamolta saatua prosessi- sekä analysaattoridataa. Tutkimusosa sisältää myös kuvauksen yleisimmistä analysaattoridatan tiedonsiirtomenetelmistä.

Työn kokeellinen osa sisältää kolme pääosaa. Ensimmäisessä osassa havainnollistetaan, kuinka tiettyjen laadunvalvontasuureiden estimointitarkkuutta voidaan parantaa, kun analysaattori-informaatiota sekä prosessidataa hyödynnetään mallinnuksessa. Toisessa osassa esitellään analysaattoritiedonkeruujärjestelmä, joka esittää myös ADI sovittemen ABB online analysaattorille. Tämä OPC UA palvelimeen perustuva sovitin tarvittiin, sillä tutkittu jalostamoyksikkö käyttää ABB:n laitteita laadunvalvonnassa. Viimeisessä osiossa esitellään laskentapalvelu kemometrisiin laskentatarpeisiin. Tämä palvelu implementoi datakäsittelyketjun prosessi- sekä analysaattorimittauksille. Palvelu hyödyntää OPC UA protokollaa ja edellä luotua tiedonkeruujärjestelmää.

**Avainsanat** OPC UA, Analysaattorit, Kemometria, Dataintegraatio

# Preface

This master's thesis was authored at the Technology and Product Development of Neste Jacobs in Porvoo during the time between the June and the December 2014. This thesis was accomplished in cooperation with Neste Oil and it was supervised by professor Sirkka-Liisa Jämsä-Jounela from Aalto University.

I want to thank my instructors D.Sc. Mikko Vermasvuori and M.Sc. Lauri Haapanen for all the guidance they gave me throughout the process. I greatly appreciate the whole Industrial ICT, Dynamic Simulation and Real Time Optimizations teams at Neste Jacobs for helping me and providing great environment to work in. I also want to thank my professor Sirkka-Liisa Jämsä-Jounela for all the guidance she gave me during my studies at Aalto University. In addition I want to thank D.Sc. Kari Aaljoki who showed interest towards my thesis and for helpful feedback he gave me. Special thanks go to D.Sc. Taito Väänänen at Neste Oil Oyj who provided me the testing data used in this thesis.

Finally I want to thank my mother and sister for all the support they gave me. I also want to thank my friends and colleagues for the inspiring company.

I hope you enjoy reading this thesis.

Porvoo 27.03.2015

Markus Sintonen

## Table of Contents

## NOMEANCLATURE

### Symbols

| | |
|---|---|
| $F$ | F-statistic for F-test |
| $f_{a,b}(\alpha)$ | Value of F-distribution with $a$ and $b$ degrees of freedom at $\alpha$ critical value |
| $K$ | Number of the principal components or latent variables in a multivariate model |
| $k$ | Index of the principal component of the latent variable |
| $L$ | Number of the distinct eigenvalues of a matrix |
| $M$ | Number of the variables |
| $m$ | Index of a variable |
| $N$ | Number of the samples |
| $n$ | Index of a sample |
| $N_v$ | Number of the cross validation experiments |
| $N_{rep}$ | Number of the repetitions in the Monte-Carlo cross validation experiments |
| $R^2$ | Coefficient of determination |
| $R_R$ | Ratio of sum of squares of the prediction error and residuals |
| $SPE_x$ | Squared prediction error of explanatory variable |
| $SPE_{x,lim}$ | Control limit for the squared prediction error of the explanatory variable |
| $T^2$ | Hotelling $T^2$-statistic |
| $T_{lim}^2$ | Control limit of Hotelling $T^2$-statistic |
| $v$ | Energy state of the harmonic or anharmonic oscillator |
| $x_{nm}$ | n:th sample of the m:th explanatory variable |
| $\bar{x}_m$ | Mean of the m:th explanatory variable |
| $\hat{x}_{nm}$ | n:th sample estimate of the m:th explanatory variable |
| $y_{nm}$ | n:th sample of the m:th response variable |
| $\hat{y}_{nm}$ | n:th sample estimate of the m:th response variable |
| $\bar{y}_m$ | Mean of the m:th response variables |
| $w_{km}$ | Weight of the k:th latent variable corresponding to the m:th variable |
| $\|\cdot\|$ | Euclidean norm operator |

## Vectors and matrices

| | |
|---|---|
| $\boldsymbol{b}$ | Vector of the regression coefficients |
| $\boldsymbol{c}_{T^2}$ | Contribution of each explanatory variable to the $T^2$-statistic |
| $\boldsymbol{c}_{SPE}$ | Contribution of each explanatory variable to the $SPE$-statistic |
| $\boldsymbol{e}$ | Residual vector |
| $\boldsymbol{p}_k$ | Explanatory variables loading vector for the k:th latent variable |
| $\boldsymbol{q}_k$ | Response variables loading vector for the k:th latent variable |
| $\boldsymbol{t}_k$ | Explanatory variables score vector for the k:th latent variable |
| $\boldsymbol{u}_k$ | Response variables score vector for the k:th latent variable |
| $\boldsymbol{w}_k$ | Weight vector for the k:th latent variable |
| $\overline{\boldsymbol{x}}$ | Mean of each explanatory variable from sample population |
| $\widehat{\boldsymbol{x}}$ | Estimate of each explanatory variable |
| $\boldsymbol{x}_m$ | m:th explanatory variable vector |
| $\boldsymbol{x}_{m,C}$ | Mean centered m:th explanatory variable |
| $\boldsymbol{x}_{m,AS}$ | Autoscaled m:th explanatory variable |
| $\boldsymbol{x}_{m,MSC}$ | Multiplicative scatter corrected m:th explanatory variable |
| $\boldsymbol{x}_{m,SNV}$ | Standard normal variate of the m:th explanatory variable |
| $\boldsymbol{x}'_m$ | First backward finite difference vector of the m:th explanatory variable |
| $\boldsymbol{x}''_m$ | Second backward finite difference vector of the m:th explanatory variable |
| $\boldsymbol{x}_n$ | n:th sample vector of the explanatory variables |
| $\boldsymbol{x}_{ref}$ | Reference sample for a multiplicative scatter correction |
| $\overline{\boldsymbol{y}}$ | Mean of each response variable from sample population |
| $\widehat{\boldsymbol{y}}$ | Estimates of each response variable |
| $\boldsymbol{y}_m$ | m:th response variable vector |
| $\boldsymbol{y}_n$ | n:th sample vector of the response variables |
| $\boldsymbol{B}$ | Regression matrix |
| $\boldsymbol{E}$ | Residual matrix of the explanatory variables |
| $\boldsymbol{F}$ | Residual matrix of the response variables |
| $\boldsymbol{G}$ | Residual matrix from regression between $\boldsymbol{T}$ and $\boldsymbol{U}$ |

| | |
|---|---|
| $\boldsymbol{P}$ | Loading matrix of explanatory variables |
| $\boldsymbol{P}_S$ | Super loading matrix |
| $\boldsymbol{Q}$ | Loading matrix of the response variables |
| $\boldsymbol{T}$ | Score matrix of the explanatory variables |
| $\boldsymbol{T}_S$ | Super score matrix |
| $\boldsymbol{U}$ | Score matrix of the response variables |
| $\boldsymbol{W}$ | Weight matrix |
| $\boldsymbol{W}_S$ | Super weight matrix |
| $\boldsymbol{S}_{\boldsymbol{XY}}$ | Cross-covariance matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices |
| $\boldsymbol{X}$ | Column matrix of the explanatory variables |
| $\widehat{\boldsymbol{X}}$ | Estimate of the explanatory variable matrix |
| $\boldsymbol{Y}$ | Column matrix of the response variables |
| $\widehat{\boldsymbol{Y}}$ | Column matrix of the response variables |

## Greek letters

| | |
|---|---|
| $\alpha$ | Confidence level |
| $\Theta$ | Sum of unexplained variance |
| $\lambda_k$ | k:th eigenvalue of a matrix |
| $\sigma_{\boldsymbol{x}}$ | Standard deviation of a vector $\boldsymbol{x}$ |
| $\Phi^{-1}(\alpha)$ | Value of the probit function with a confidence level $\alpha$ |

## Abbreviations and abbreviated variable names

| | |
|---|---|
| APC | Advanced process controller |
| BVE | Backwards variable elimination |
| COM | Component object model |
| COVPROC | Covariance procedure |
| DCS | Distributed control system |
| DSPLS | Direct scores PLS |
| EMSC | Extended multiplicative scatter correction |
| FTIR | Fourier transform infrared |
| HDF | Hydrofinishing reactor |
| HMI | Human machine interface |
| I/O | Input and output |
| IDW | Isomerization dewaxing reactor |
| IKPLS | Improved kernel PLS |
| KV100 | Kinematic viscosity at 100°C |
| LIMS | Laboratory information system |
| LV | Latent variable |
| MB-PLS | Multi-block PLS |
| MB-PLS1 | MB-PLS model with a single estimated response variable |
| MCCV | Monte-Carlo cross-validation |
| MD5 | Message-digest algorithm |
| MES | Manufacturing executing system |
| MIR | Mid-infrared |
| MLR | Multiple linear regression |
| MSC | Multiplicative scatter correction |
| NIPALS | Non-linear iterative projection to latent structures |
| NIR | Near-infrared |
| NN | Neural network |
| NRMSEP | Normalized root mean squared error of prediction |

| | |
|---|---|
| N-PLS | Nonlinear PLS |
| OPC | Open platform communications |
| OPC UA ADI | OPC UA for analyzer devices |
| OPC UA DI | OPC UA for devices |
| OPC UA | Open platform communications unified architecture |
| PC | Principal component |
| PCA | Principal component analysis |
| PLS | Projection to latent structures |
| PLS1 | PLS model with a single estimated response variable |
| PLC | Programmable logic controller |
| P-PLS | Priority PLS |
| PRESS | Prediction error sum of squares |
| RMSECV | Root mean squared error of cross-validation |
| RMSEP | Root mean squared error of prediction |
| RSS | Residual sum of squares |
| SG | Savitzky-Golay filter |
| SGD | Savitzky-Golay derivative filter |
| SNV | Standard normal variate |
| SPC | Statistical process control |
| SPE | Squared prediction error |
| SQL | Structured query language |
| SSL | Secure sockets layer |
| SVD | Singular value decomposition |
| TCP/IP | Transmission control protocol / internet protocol |
| UVE | Uninformative variable elimination |
| VHVI | Very high viscosity index |
| VIP | Variable importance in projection |
| XCF | Cross-correlation function |

# LITERATURE PART

## 1. Introduction

Analyzers are instruments that can conduct chemical analysis of individual samples or sample streams. Throughout the chemical industries these instruments are used for a wide range of different applications including product quality monitoring, process control and process optimization. A number of different analyzer devices have been developed based on different physical principles and include, for example, infrared spectrometers, mass spectrometers and Raman spectrometers. This thesis focuses on information systems and multivariate calibration related to the spectroscopic instruments producing measurements in forms of spectra.

Measurement and status related information produced by the instruments are normally only available locally from the physical device. Measured spectrum is not usually directly useful in quality and process control applications. As a result the spectrum is typically processed locally on the analyzer device based on configuration and calibration of the device. The final measured quantity is then passed on from the instrument to the higher level automation systems e.g. the plant information system. Spectra and status related information are not readily accessible from the plant information systems; instead, the information is only contained in the devices in proprietary formats. However, some of the more advanced analyzer devices can make this information available through proprietary protocols which are remotely accessible through vendor specific information systems. [1] [2]

Large industrial applications might use wide range of different types of analyzer devices from multiple manufacturers. This creates a situation where all the information produced by the different analyzers is spread out over the plant floor and is only accessible though proprietary protocols. Consequently only the most basic quantity estimates produced by the devices are readily available from the higher level information systems. It would be valuable for the end users to have a single access point that provides a united view on the plant analyzer information. Such a situation would allow more convenient access to the analyzer information when compared to being able to only obtain data locally from the device. This kind of data acquisition has a wide range of different application e.g. spectral measurements can be combined with other types of plant data to create added value to the quality control, which is also used as a case study in this thesis. Analyzer data integration has also applications in status monitoring of the plant analyzer instruments, for example, this can be used for the early detection of analyzer calibration deterioration. In this type of application the plant engineer would get notification from the analyzer information system that recalibration or corrective actions are required to keep the specific analyzer instrument operational.

This thesis examines one particular application where plant measurement and analyzer data are integrated behind a single access point. Multivariate methods are then used to create a mathematical model that combines these information sources to yield more accurate estimates of specific product properties. Multivariate methods are widely used in spectroscopic data analysis and these data-driven methods are commonly used in the field of chemometrics where chemically relevant information is extracted from datasets produced during chemical experiments. Chemometrics utilize multivariate-, classification-, pattern recognition- and clustering mathematical methods, which are used to determine the structure and underlying relationships within the chemical system. Properties and behavior of the chemical system can be predicted in chemometric applications by modelling of the system. Datasets used in these applications are often very large and involve hundreds to thousands of variables. Similarly spectroscopic instruments produce vast amounts of data in form of a spectrum with each frequency or wavelength in a spectrum representing one variable in the dataset, hence multivariate methods are valuable tools for the analysis of these types of large datasets.

Recent standardized automation related information models and data transfer protocols allow access to the device information regardless of the manufacturer specific implementations. These specifications are utilized in this thesis to access the plant analyzer devices and other information sources. Firstly data acquisition software is created to allow access to the plant data from a single access point. Additionally an in-house chemometric calculation platform is developed to utilize information from the data acquisition system to refine the analyzer and plant information into more useful property estimates used for product quality control.

## 2. Basic principles of a vibrational spectroscopy

Vibrational spectroscopic techniques are based on molecular vibrations caused by electromagnetic excitation of the analyte. These techniques are divided into three categories by frequency range and physical principle: NIR (near-infrared), MIR (mid-infrared) and Raman. NIR and MIR techniques rely on the absorption of electromagnetic radiation at different frequencies, whereas the Raman technique relies on the scattering of the electromagnetic radiation. Near-infrared light sources operate in the 4000-12,500 cm$^{-1}$ wavenumber range whilst both mid-infrared and Raman spectrometers operate in the 200-4000 cm$^{-1}$. [3]

The absorption of a photon causes a change in the vibrational state of a molecule. Vibration of diatomic molecules, for example, can be approximated to harmonic oscillators where the vibrating masses are connected by a spring. This leads to a parabolic energy potential of the oscillator where the system can only be in a discrete energy state. These states can be obtained after the system is treated as a quantum mechanical system and the Schrödinger equation is applied to the energy function. Energy states (v) and harmonic oscillator potential energy are displayed in Figure 1. Energy state transitions within a harmonic oscillator are only allowed when the state changes by one ($\Delta v = \pm 1$) and these transitions are known as fundamental transitions.

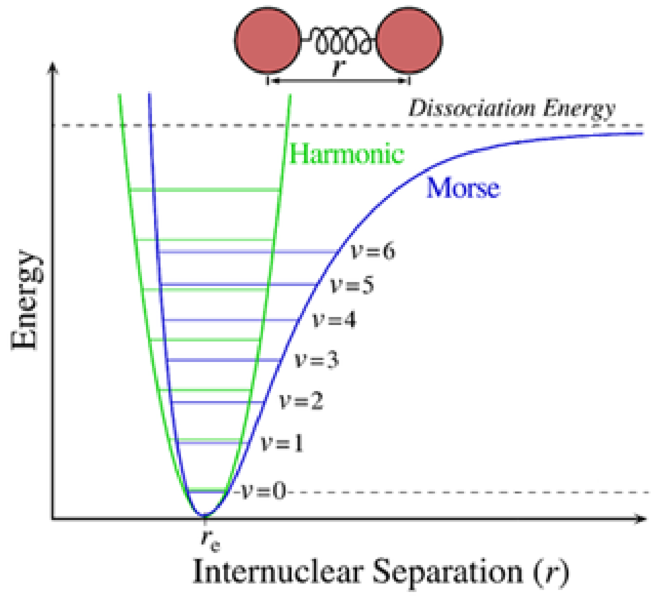


Figure 1. Potential energy of a harmonic oscillator (green) and Morse potential (blue) with energy states (*v*) displayed [4]

Harmonic approximation does not hold for larger vibrational amplitudes. This is due to repulsive forces between the vibrating atoms and the possibility of dissociation when the bond is excessively extended. Therefore the energy function of an anharmonic oscillator is not parabolic, and can be approximated using a Morse potential as shown in Figure 1. When a molecule dictated by the Morse potential energy function is treated with the Schrödinger equation, one obtains vibration state transitions that are much more versatile than those of the harmonic oscillator. Energy states modelled by the Morse potential have a higher number of possible discrete integer transitions ($\Delta v=\pm1,\pm2,\pm3$, etc.) with transitions greater than one called overtone transitions. These transitions cause overtone vibrations and they are observed as overtone bands on the absorbance spectrum. Bands caused by overtones are much weaker and broader than fundamental bands and overtone bands get weaker and weaker as a transition distance between energy states increases. This is due to the fact that overtone excitations are statistically less likely to happen than fundamental excitations. Bands caused by overtone excitations also move to higher frequencies as the distance of the transition increases.

Overtone and fundamental bands are both caused by changes in vibration energy states. In addition there is a third class of vibrational bands, knows as *combination bands* which are caused by simultaneous excitation of more than one vibrational mode. [5]

The MIR frequency area is dominated by fundamental bands whereas at NIR frequencies the overtone and combination bands predominate. Structural selectivity of spectral bands inside the NIR region is much lower than those inside the MIR frequency region as the MIR region is dominated by fundamental bands that are more structurally selective. Additionally overtone and combination bands are generally highly overlapped within the NIR frequency range, therefore interpretation of the NIR spectra is much more difficult when compared to the MIR spectra. NIR applications only became viable after development of chemometric methods in the 70s which led to the breakthrough of NIR applications in different industries. [5] Applications of NIR analyzers include: chemical industries where product quality or emissions are monitored online [6]; medical applications to monitor, for example, blood oxygen or hemoglobin of a patient [7] and agricultural for non-destructive testing of crop quality. [8] Applications exist even in astronomy where NIR analyzers are used to measure composition of molecular clouds in distant star systems. [9]

The main advantages of NIR techniques over MIR techniques include ease of sample handling, possibility of using large samples and possibility to use fiber optics with the NIR equipment. Fiber optics are used with NIR analyzers in the chemical industries to remotely monitor the sampling points which can be for example product streams [5]. These advantages are the main reasons why NIR analyzers are in such a dominant position in different industries when compared to other spectroscopic instruments. [3]

IR analyzers are mostly Fourier transform infrared based instruments (FTIR). This method is based on an interference pattern in a time domain that is Fourier transformed into the final spectrum in a frequency domain. An interference pattern is formed by the Michaelson interferometer, which (shown in Figure 2) is based on a stationary and a moving mirror with a beam splitter. A recombined interfered IR beam is shone through the analyte and a pattern is recorded by a detector. The advantages of FTIR are high wavenumber stability and resolution compared to other interferometer based analyzers. NIR analyzers are also commonly based on FTIR, although recently more sophisticated devices have also been developed and include micro-electro-mechanical-systems and acousto-optic-tunable-filters. This has led to the miniaturization of NIR analyzers since these devices require no moving parts. [3] [5]

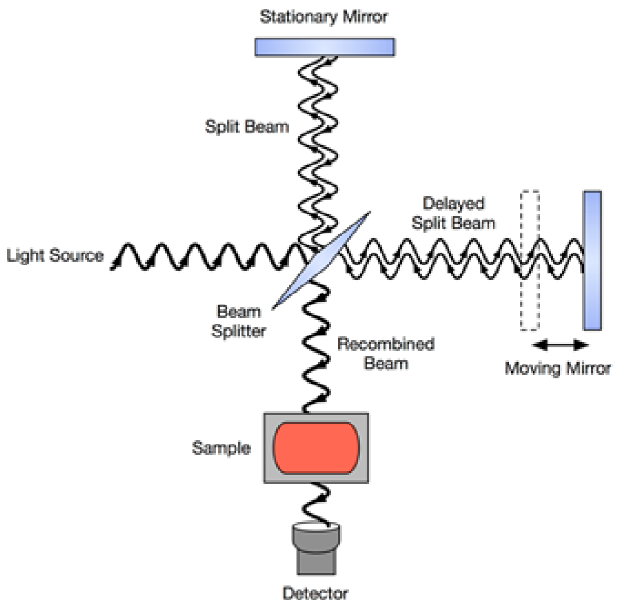


Figure 2. Schematics of Fourier transformed infrared spectroscopy based on the Michelson interferometer [10]

# 3. Data processing in spectroscopic applications

All spectroscopic instruments produce data samples in the form of spectra. Vibrational spectroscopy spectra represent absorbance of electromagnetic radiation at different frequencies and this thesis focuses on the processing of spectra produced by instruments relying on infrared radiation. All the methods outlined in this work can also be applied to spectra produced by other types of instruments.

## 3.1 Spectrum preprocessing

Preprocessing of the raw NIR spectrum is an essential step in chemometric modelling. Raw spectra might be affected by solid particles, water or bubbles that can cause scattering of the NIR beam and result in the drift of the spectrum baseline that lowers the performance of the model. Spectra can also be affected by a constant, first- or higher-order additive error in the baseline, by measurement noise, varying sample cell temperature and varying analyzer light source intensity. Preprocessing methods try to compensate for the baseline and the noise related problems, however, variation in the sample temperature and problems with the analyzer lamp are more difficult to take into account using preprocessing. Signal noise is reduced using signal smoothing techniques, but these methods have the inherent problem of information loss. Multivariate models generated by preprocessed data are also usually less complex models than those generated by noisy data. Baseline problems can be compensated by de-trending or by taking the first- or higher-order derivatives of the spectrum. In addition, other techniques including Multiplicative Scatter Correction (MSC) and Standard Normal Variate (SNV) have been developed for baseline correction. Normalization of the data is also important prior to applying multivariate methods and numerous preprocessing methods have been proposed for spectral data. This poses significant challenges for determining the right methods and no general rules are available. As a result the choice of suitable preprocessing methods is highly application specific. [11] [12]

### 3.1.1 Data normalization

Data centering is the most common preprocessing procedure for the spectral data and can be a sufficient preprocessing step in a case where there are no problems related to the baseline or measurement noise. Data centering is performed by subtracting sample average from each variable as shown in Equation (1). [11]

$$\boldsymbol{x}_{m,C} = \boldsymbol{x}_m - \bar{x}_m = \boldsymbol{x}_m - \frac{1}{N}\sum_{n=1}^{N} x_{nm} \tag{1}$$

where *n* is index of the sample

*m* is index of the wavenumber

*N* is number of samples

Scaling of the data is essential when variables have highly different magnitudes as multivariate methods give equal emphasis for each variable only if they have been properly scaled. As a consequence, autoscaling is usually performed prior to the application of multivariate methods. This standardization is applied by dividing a centered variable by their standard deviation. [11] In this thesis unbiased estimates of the standard deviations are used as shown in the next equation:

$$\boldsymbol{x}_{m,AS} = \frac{\overline{\boldsymbol{x}}_m}{\sigma_{x_m}} = \frac{\overline{\boldsymbol{x}}_m}{\sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{nm} - \bar{x}_m)^2}} \tag{2}$$

where $\sigma_{x_m}$ is an unbiased standard deviation estimate of the m:th wavenumber variable

### 3.1.2 Scatter correction methods for the spectral data

Most of the preprocessing techniques developed for spectroscopic methods try to compensate for the undesired effect of light scattering as it causes a baseline drift on the measured spectrum. [5] This systematic variation is undesired since it affects performance of multivariate methods so MSC and SNV are the most commonly applied preprocessing approaches used for scatter correction. Data normalization, as presented above, can be also used for correcting an additive baseline drift, as can detrending for correcting linear or higher-order multiplicative baseline shifts.

The SNV method is closely related to the autoscaling of the data and the following equation is used to determine the SNV corrected data. [12]

$$\boldsymbol{x}_{m,SNV} = \frac{\boldsymbol{x}_m - \bar{x}_n}{\sigma_{x_n}} \tag{3}$$

Equation (3) shows how SNV uses an average of a sample spectrum $\bar{x}_n$ instead of the wavenumber variable $\bar{x}_j$ average. Similarly, the standard deviation of the sample spectrum $\sigma_{x_n}$ is used instead of variables standard deviation.

MSC is another commonly used scatter correction method and it has the same form as autoscaling and SNV. MSC uses correction coefficients for correcting additive and multiplicative contributions as detailed in Equation (4):

$$\boldsymbol{x}_{n,MSC} = \frac{\boldsymbol{x}_n - a_n}{b_n} \tag{4}$$

where $a_n$ is coefficient for correcting additive contribution for a sample n

$b_n$ is coefficient for correcting multiplicative contribution for a sample n

The above coefficients are determined for each measured spectrum sample separately and are obtained by the first-order least squares fitting of the measured spectra and a reference spectrum as shown in Equation (5).

$$\boldsymbol{x}_n = a_n + b_n\boldsymbol{x}_{ref} + \boldsymbol{e}_n \tag{5}$$

A reference spectrum $\boldsymbol{x}_{ref}$ is usually an average of each wavenumber used in the calibration dataset.

Both MSC and SNV have their advantages and disadvantages with the most suitable method depending on the application. SNV is sensitive to signal noise and outliers whereas the MSC method is sensitive to the reference spectrum. [12]

A more advanced methods have also been proposed and include more robust versions of MSC and SNV. The robust version of SNV uses median values of the spectrum instead of mean values and the variability of the data is estimated using the median absolute deviation (mad) instead of the standard deviation. Formulation of the robust SNV is shown in the following Equation (6). [13]

$$\boldsymbol{x}_{m,SNV} = \frac{\boldsymbol{x}_m - median(\boldsymbol{x}_n)}{mad(\boldsymbol{x}_n)} = \frac{\boldsymbol{x}_m - median(\boldsymbol{x}_n)}{1.4826\, median(|\boldsymbol{x}_m - median(\boldsymbol{x}_n)|)} \tag{6}$$

Similarly the robust autoscaling can be performed using the median and median absolute deviation. Verboven *et al.* have also proposed a robust version of MSC where the median reference spectrum is used instead of the average reference spectrum. They also proposed using the robust least trimmed squared regression (LTS) instead of the normal least squares regression and such robust versions of the baseline correction are suitable for cases when a spectrum contains outliers. [13] In addition the Extended MSC (EMSC) methods have been proposed. These are augmented versions of MSC where the second-order polynomials are used instead of the first-order polynomials. Expansions also contain weights for wavenumber-axis dependencies and inclusion of a priori information about spectral interferents. For example, the least amount of weight can be given to certain, unwanted, wavelength regions in the spectrum, however in real applications the choice of these parameters might be a tedious task. [12]

MSC and SNV both try to compensate for the influence of light scattering inside a specific sample cell. Therefore these methods do not necessarily handle correctly baseline differences between instruments. Therefore the transfer of a multivariate model and preprocessing parameters to a different instrument might result in the deterioration of the estimation performance. Therefore several instrument standardization techniques have been proposed to overcome the model transfer related problems. [5]

### 3.1.3 Spectral derivatives

Derivatives are used both to bring out overlapping bands in the spectrum and to compensate for a baseline drift. The first derivative removes any additive baseline shift whilst the second derivative also eliminates the effect of a multiplicative baseline drift. Basic implementation of the first and second order derivatives can be performed by the finite backward differences:

$$\boldsymbol{x}'_m = \boldsymbol{x}_m - \boldsymbol{x}_{m-1} \tag{7}$$

$$\boldsymbol{x}''_m = \boldsymbol{x}_m - 2\boldsymbol{x}_{m-1} + \boldsymbol{x}_{m-2} \tag{8}$$

Taking finite differences has a fundamental problem with noisy signals since it greatly amplifies noise in the final signal. However, a noisy signal can be smoothed before taking the finite difference by using one of the many smoothing techniques available like, for example, a simple moving window average filter. [11] [12] Another common method used for spectral data is the Savitzky-Golay (SG) filter, which is based on the least squares fitting of polynomials over a specific window. Smoothness of the final signal can be adjusted by changing the window size and the degree of the polynomial. The derivative can be also calculated directly from the fitted polynomial and is known as the Savitzky-Golay derivation method (SGD). Overall the order of the SG-filter polynomial dictates the maximum order of the derivative and the smoothness of the filtered signal. [12]

Research by M. Faber, however, demonstrated that taking derivatives could have a very negative impact on the predictive performance of multivariate methods. The author used finite differences for calculating derivatives and evaluated the estimation performance based on multivariate sensitivity. He also stated that signal smoothing would not influence the situation since multivariate methods have an inherit ability to filter out the impact of the noise. It was also shown that viability of spectral derivatives is application specific and different results have been reported in literature. [14]

## 3.2 Multivariate modelling of the spectral data

Spectrometric multivariate data analysis falls into two categories: multivariate calibration and multivariate diagnostic. Calibration models are created in order to estimate the values of some specific property of an analyte using the measured spectrum. These models can also be utilized in detecting problems with the spectrometric instrument or in the measured system. Research of multivariate models for extracting chemically relevant information from measurement data falls into the discipline of chemometrics and the most prominent methods used include principal component analysis (PCA) and projection to latent structures (PLS). [5] Numerous extensions of these and more advanced methods have also been proposed and comprise of various PLS extensions, for example Multi-block PLS (MBPLS), Priority PLS (PPLS) and nonlinear PLS (NPLS). [11] Other, more advanced, methods include utilization of neural networks (NN), whilst combinational methods for example, when PLS scores are used as NN inputs, also exist. [15], [16]

Many PLS variants have been developed in which the explanatory and response variables are broken into separate meaningful blocks. A dataset can be divided into blocks based on instrument types or by physical locations of the measurements. Process measurement variables might be valuable additions to the input variable set when creating chemometric models. The effect of these few additional variables would be insignificant compared to thousands of wavelength variables, if they are directly added to the explanatory variable matrix. Therefore, models that give equal importance to the variable sets are needed when data comes from different types of sources. This thesis outlines a priority PLS (P-PLS) model where the data blocks are treated one at a time and multi-block PLS (MB-PLS) where the data blocks are handled simultaneously. In addition, some adaptive schemes for chemometric models are outlined which can take changes in the operational conditions into account.

### 3.2.1 Multiple linear regression

Multiple linear regression (MLR) is the most traditional method for building a relationship between explanatory variables and a response variable. MLR produces a calibration model using the classical least-squares method. Spectroscopic measurements produce data in the form of an absorbance or transmittance where a single wavenumber of the measurement represents one variable. This dataset is organized into a matrix and is denoted as ***X***, whilst the response variable ***y***, might be for example, a vector of measured concentrations. MLR then finds the best possible fit of a vector ***b*** that relates these two properties together:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{b} \tag{9}$$

The objective is to find a regression vector ***b*** that minimizes a sum of the squared errors between the measured and estimated response variables:

$$J = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}) \tag{10}$$

The solution to the least squares problem is obtained by the following equation:

$$\boldsymbol{b} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{11}$$

A response variable ***y*** can be then estimated from a new measurement by using Equation (9).

The inversion of $\boldsymbol{X}^T\boldsymbol{X}$ makes solving of this equation difficult as the matrix is not invertible if ***X*** has more columns than rows. Moreover, collinearity of the matrix ***X*** makes the inversion problem ill-conditioned and the spectrum sample matrix ***X*** is usually highly collinear due to the interaction between the adjacent wavenumbers. [5] This collinearity can be handled by dimensionality reduction techniques that create orthogonal projections of the data matrices. Some of these methods are presented in the following paragraphs.

### 3.2.2 Principal component analysis and -regression

Principal component analysis (PCA) is a dimensionality reduction technique for multidimensional data. The objective of PCA is to find linearly uncorrelated principal components (PC) that best describe the underlying structure of the data and these components are selected so that the loss of information is minimized. The first principal component accounts for maximum amount of variability in the data, whilst subsequent components increase the descriptiveness of the variation within the data. The number of the used principal components in a multivariate model is evaluated so that the underlying structure of the data is captured and noise is filtered out. Principal components can be obtained by singular value decomposition (SVD) of a data matrix:

$$\boldsymbol{Z} = \frac{1}{\sqrt{N-1}}\boldsymbol{X} \tag{12}$$

$$\boldsymbol{Z} = \boldsymbol{USV}^T \tag{13}$$

where ***U*** is a matrix with columns contain eigenvectors of $\boldsymbol{ZZ}^T$

***S*** is a matrix with diagonal elements containing singular values of ***X*** which are square roots of the eigenvalues of $\boldsymbol{Z}^T\boldsymbol{Z}$

***V*** is a matrix with columns contain eigenvectors of $\boldsymbol{Z}^T\boldsymbol{Z}$

Each variable in the data matrix ***X*** must be autoscaled prior to the application of Equations (12) and (13).

It can be seen that the matrix $\boldsymbol{Z}^T\boldsymbol{Z}$ is an estimate of the covariance matrix of ***X*** since**:**

$$\boldsymbol{Z}^T\boldsymbol{Z} = \left(\frac{1}{\sqrt{N-1}}\boldsymbol{X}\right)^T\left(\frac{1}{\sqrt{N-1}}\boldsymbol{X}\right) = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X} \tag{14}$$

Therefore the columns of ***V*** contain the eigenvectors of the ***X***'s covariance matrix and are known as the principal components of ***X***. In contrast the diagonal matrix ***S*** holds singular values (also known as principal values) in a decreasing order. Squares of these correspond to the eigenvalues of the covariance matrix. The corresponding eigenvectors in ***U*** and ***V*** are in decreasing order of variance explained in ***X***. Some of these principal components describe the relevant information related to the estimated quantity, whilst other components only describe random measurement noise or irrelevant structures within the data. Only those principal components describing the structure of the data should be chosen for the final model and the others discarded. The chosen components should describe most of the relevant variance in the data and these are the first eigenvectors in the matrix ***V*** corresponding to the largest eigenvalues. [5] Various methods have been proposed on how to choose the correct number of the principal components included in the model. [17] [18] It is highly important that an optimal model order is selected, as a too high model order causes model overfitting, which means that the model takes irrelevant information into account thus decreasing the estimation performance of the model. Methods used for determining the correct model order are outlined in more detail in Chapter 3.3 Assessment of the model complexity.

PCA is a dimensionality reduction technique that reduces the dimensionality of the data matrix. A compressed representation of the data matrix is obtained by multiplying it with a loading matrix ***P***. This compressed data matrix is called a score matrix ***T***:

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{P} \tag{15}$$

where ***X*** is the data matrix

***P*** is a loading-matrix containing the selected principal components as columns in a decreasing order of the variance (as explained above)

This compressed representation of ***X*** can be then decompressed into an estimate of ***X***:

$$\widehat{\boldsymbol{X}} = \boldsymbol{T}\boldsymbol{P}^T \tag{16}$$

This can be used to test how well the model fits the new measurement dataset. This is useful for model validation and process monitoring purposes.

Principal components can also be used to regress a response variable on the explanatory variables - a method known as principal component regression (PCR). A regression coefficient vector ***b*** is obtained in a similar manner to Equation (11) with the ordinary MLR method. Calculation of the regression coefficients in PCR uses the score-matrix ***T*** instead of using the data matrix ***X***:

$$\boldsymbol{b} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{y} \tag{17}$$

This regresses the response variable on the PCA score-matrix ***T***. An estimate for the response variable can be obtained for a new measurement by using Equation (9) with the score-matrix ***T*** in the place of the data matrix ***X***. Regression of the response variables can be used to estimate the desired properties and in addition, the estimate can be used to evaluate the optimal number of the principal components. PCR addresses problems related to data collinearity. This problem caused Equation (11) to be unsolvable with the MLR method when data was collinear. PCR solves this problem by projecting the dataset on space spanned by the orthogonal principal components. Orthogonality of the matrix ***T*** ensures that Equation (17) is solvable. [17]

### 3.2.3 Projection to latent structures

Projection to latent structures (PLS) is closely related to PCA and MLR. PCA captures the maximal variance in explanatory variables whereas MLR finds the maximum correlation between the response and the explanatory variables. PLS is a method that finds a balance between these two as it attempts to simultaneously optimize the explained variance and the covariance between the explanatory and the response variables. PLS uses latent variables (LV) instead of principal components of PCA and these can be interpreted as transformed versions of the principal components having the optimal capabilities to estimate values of the response variables. PCR and MLR methods regress only a single response variable on the explanatory variables whereas PLS can regress multiple response variables on the explanatory variables. The underlying model of PLS is described in Equations (18)-(20): [11]

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} \tag{18}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^T + \boldsymbol{F} \tag{19}$$

where ***T*** and ***P*** are the score- and loading-matrices of the explanatory variables ***X*** respectively

***U*** and ***Q*** are the score- and loading-matrices of the response variables ***Y*** respectively

***E*** and ***F*** are error terms

Equations (18) and (19) describe the inner relationship of the data. The outer relationship is formed by regressing the ***Y*** score matrix on the ***X*** score matrix:

$$\boldsymbol{U} = \boldsymbol{T}\boldsymbol{C} + \boldsymbol{G} \tag{20}$$

where ***C*** is a diagonal matrix containing regression coefficients for regressing ***U*** on ***T***

***G*** is residual from regression between ***T*** and ***U***

The error term matrices ***E, F*** and ***G*** should only represent noise or irrelevant information for estimating the desired property when the model order is chosen correctly. Non-linear iterative projection to latent structures (NIPALS) is a classical algorithm for PLS decomposition and this algorithm is outlined next.

A. Preprocess ***X*** and ***Y***. Set $\boldsymbol{X}_1 = \boldsymbol{X}$, $\boldsymbol{Y}_1 = \boldsymbol{Y}$ and $k = 1$

B. Choose an initial score vector ***u*** from one of the ***Y*** columns

C. Calculate the ***X*** weight vector ***w*** for the latent variable *k*:

$$\boldsymbol{w}_k = \frac{\boldsymbol{X}_k^T \boldsymbol{u}_k}{\|\boldsymbol{X}_k^T \boldsymbol{u}_k\|} \tag{21}$$

D. Calculate the ***X*** score vector ***t*** for the latent variable *k*:

$$\boldsymbol{t}_k = \boldsymbol{X}_k \boldsymbol{w}_k \tag{22}$$

E. Calculate the ***Y*** loading vector ***q*** for the latent variable *k*:

$$\boldsymbol{q}_k = \frac{\boldsymbol{Y}_k^T \boldsymbol{t}_k}{\boldsymbol{t}_k^T \boldsymbol{t}_k} \tag{23}$$

F. Calculate the ***Y*** score vector the ***u*** for latent variable *k*:

$$\boldsymbol{u}_k = \frac{\boldsymbol{Y}_k^T \boldsymbol{q}_k}{\boldsymbol{q}_k^T \boldsymbol{q}_k} \tag{24}$$

G. Test the ***u*** score vector for convergence using Equation (25). Go to H if ***u*** has converged, otherwise return to the step C.

$$\frac{\|\boldsymbol{u}_{k-1} - \boldsymbol{u}_k\|}{\|\boldsymbol{u}_k\|} < \varepsilon \tag{25}$$

where $\varepsilon$ is a small convergence tolerance

H. Calculate the ***X*** loading vector ***p*** for the latent variable *k*:

$$\boldsymbol{p}_k = \frac{\boldsymbol{X}_k^T \boldsymbol{t}_k}{\boldsymbol{t}_k^T \boldsymbol{t}_k} \tag{26}$$

I. Deflate data matrices by removing calculated latent variables contribution from the ***X*** and ***Y*** matrices using the Equations (27) and (28).

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - \boldsymbol{t}_k \boldsymbol{p}_k^T \tag{27}$$

$$\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k - \boldsymbol{t}_k \boldsymbol{q}_k^T \tag{28}$$

J. Stop if all latent variables have been calculated. Otherwise let $k = k + 1$ and return to the step B

The decomposition matrices ***T, P, U, Q*** and the weight matrix ***W*** are formed by appending the corresponding computed vectors to these matrices. A single iteration step of the NIPALS algorithm is illustrated in the following arrow scheme diagram:

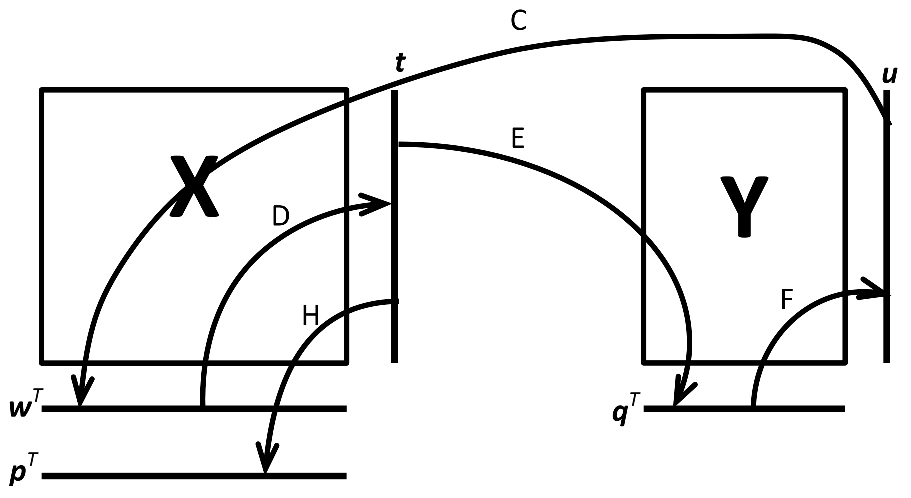


Figure 3. Iteration steps of the NIPALS algorithm. [11]

The response variable is estimated from a new measurement as follows:

$$\widehat{\boldsymbol{y}} = \overline{\boldsymbol{y}} + (\boldsymbol{x} - \overline{\boldsymbol{x}})\boldsymbol{B} \tag{29}$$

where $\overline{\boldsymbol{y}}$ is a column vector of the response variables averages used for mean centering in calibration

$\overline{\boldsymbol{x}}$ is a column vector of explanatory variables averages used for mean centering in calibration

$\boldsymbol{B}$ is a PLS regression matrix

$\boldsymbol{x}$ is a column vector of the new measurements autoscaled in the same way as calibration data

The PLS regression matrix is obtained using the following formula: [11]

$$\boldsymbol{B} = \boldsymbol{W}(\boldsymbol{P}^T\boldsymbol{W})^{-1}\boldsymbol{Q}^T \tag{30}$$

NIPALS is the classical algorithm used for obtaining PLS decomposition matrices and numerous other methods have been developed for calculating these matrices which Andersson surveyed in his study. [19] The standard NIPALS method produced very accurate estimates, but it was also most computationally intensive, whilst some of the newer algorithms have better computational performance characteristics, but some of them are also less accurate. The author determined that the so called improved kernel PLS (IKPLS) and direct scores PLS (DSPLS) algorithms were as accurate as the NIPALS algorithm. In addition these newer methods were 2-4 times faster than the standard NIPALS algorithm. The DSPLS algorithm was only suited for a single response variable, but the IKPLS method was developed for handling multiple response variables simultaneously as in the NIPALS method. [19] The NIPALS algorithm is computationally expensive because of the explanatory and response variables deflation step in Equations (27) and (28). The IKPLS algorithm overcomes this problem by only deflating the cross-covariance matrix of ***X*** and ***Y***. The IKPLS algorithm is outlined below:

A. Preprocess ***X*** and ***Y***, set *k = 1*. Compute the cross-covariance matrix: $\boldsymbol{X}^T\boldsymbol{Y}$

$$\boldsymbol{S}_{XY,k} = \boldsymbol{X}^T\boldsymbol{Y} \tag{31}$$

B. Calculate the largest eigenvector $\boldsymbol{v}_k$ of $\boldsymbol{S}_{XY,k}^T\boldsymbol{S}_{XY,k}$

C. Determine the ***X*** weight vector ***w*** for the latent variable *k*:

$$\boldsymbol{w}_k = \frac{\boldsymbol{S}_{XY,k}\boldsymbol{v}_k}{\left\|\boldsymbol{S}_{XY,k}\boldsymbol{v}_k\right\|} \tag{32}$$

D. Calculate the vector $\boldsymbol{r}_k$ for the latent variable *k*:

$$\begin{aligned} &\boldsymbol{r}_1 = \boldsymbol{w}_1 \\ &\boldsymbol{r}_k = \boldsymbol{w}_k - \boldsymbol{p}_1^T\boldsymbol{w}_k\boldsymbol{r}_1 - \boldsymbol{p}_2^T\boldsymbol{w}_k\boldsymbol{r}_2 - \cdots - \boldsymbol{p}_{k-1}^T\boldsymbol{w}_k\boldsymbol{r}_{k-1}, \ \ k > 1 \end{aligned} \tag{33}$$

E. Calculate the ***X*** score vector $\boldsymbol{t}_k$ for the latent variable *k*:

$$\boldsymbol{t}_k = \boldsymbol{X}\boldsymbol{r}_k \tag{34}$$

F. Determine ***X*** and ***Y*** loading vectors $\boldsymbol{p}_k$ and $\boldsymbol{q}_k$ for the latent variable *k* by using Equations (35) and (36).

$$\boldsymbol{p}_k = \frac{\boldsymbol{X}^T \boldsymbol{t}_k}{\boldsymbol{t}_k^T \boldsymbol{t}_k} \tag{35}$$

$$\boldsymbol{q}_k^T = \frac{\boldsymbol{r}_k^T \boldsymbol{S}_{XY,k}}{\boldsymbol{t}_k^T \boldsymbol{t}_k} \tag{36}$$

G. Deflate the cross-covariance matrix $\boldsymbol{X}^T\boldsymbol{Y}$ by removing the calculated latent variables contribution from it:

$$\boldsymbol{S}_{XY,k+1} = \boldsymbol{S}_{XY,k} - \boldsymbol{p}_k \boldsymbol{q}_k^T (\boldsymbol{t}_k^T \boldsymbol{t}_k) \tag{37}$$

H. Stop if all latent variables have been calculated. Otherwise let $k = k + 1$ and return to the step B

Store the obtained vectors ***w, r, t, p*** and ***q*** at each iteration to matrices ***W, R, T, P*** and ***Q*** as columns. Finally the PLS regression matrix ***B*** can be calculated:

$$\boldsymbol{B} = \boldsymbol{R}\boldsymbol{Q}^T \tag{38}$$

The IKPLS regression is performed in a similar way as in NIPALS by using Equation (29). [20]

The NIPALS and IKPLS algorithms can be simplified if there is only one response variable. In NIPALS, the steps C-G converge in a single iteration, whilst for the IKPLS algorithm, the step B is skipped and Equation (32) is calculated without the eigenvector. Moreover, it is beneficial to use a faster PLS algorithm than the classical NIPALS when dealing with large datasets. The model validation by cross-validation techniques become especially computationally expensive with large datasets since many PLS models must be tested against different validation datasets.

Accuracy of the estimate produced by the PLS model is highly dependent on the selected number of the latent variables as, for example, a too high number of the latent variables causes model overfitting. This particular problem is avoided by model cross-validation, which is used to evaluate the correct model order. These methods are presented in more detail in Chapter 3.3 Assessment of the model complexity.

### 3.2.4 Non-linear PLS regression

Modelling of a system usually starts with the assumption that the system is linear. This initial assumption can be justified in spectroscopic multivariable modelling since the underlying phenomena are based on the absorbance of electromagnetic radiation. According to the Beer-Lambert Law the absorbance is linearly dependent on the concentration and molar absorptivity of the sample. In addition the path length of the sample cell also affects absorbance linearly. Nevertheless some nonlinearity is often detected in spectral data because the response variables may be nonlinearly dependent on the explanatory variables and nonlinearities can also be observed in spectral data between the adjacent wavelengths. Many factors can contribute to the nonlinear behavior of the spectroscopic data such as instrumental sources, response dynamics, molecular interactions, concentration and temperature variations and optical scattering.

Data nonlinearities can be detected by evaluating how the estimated and observed dependent variable measurements relate to each other. A figure, where estimated measurements are plotted as a function of the observed measurements, reveals possible nonlinear behavior of the dependent variable. The score vectors ***t*** and ***u*** may also be plotted against each other for inspection of possible nonlinear behavior. [11]

Numerous different approaches have been published in the literature for handling the nonlinearities [11] [16]. One method that can be used is to add additional nonlinear variables to the explanatory variable matrix to produce that is known as an implicit non-linear latent variable regression (INLR):

$$\boldsymbol{Z} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M, \quad \boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \dots, \boldsymbol{x}_M^2, \quad \boldsymbol{x}_1\boldsymbol{x}_2, \boldsymbol{x}_2\boldsymbol{x}_3, \dots, \boldsymbol{x}_{M-1}\boldsymbol{x}_M] \tag{39}$$

Higher order terms can also be included to the augmented matrix. This method has the ability to handle nonlinearities easily and effectively, but it also has disadvantages. The size of the explanatory variable matrix multiplies because higher order terms are included to the matrix. This increases not only the computational burden but the higher order terms are also very sensitive to outliers and noise. [11]

Another approach is to use nonlinear mapping in the inner relation between the score matrices ***T*** and ***U*** of the PLS instead of the linear one, for example, a neural network or a polynomial fitting can be utilized to create the nonlinear mapping. [21] Nonlinear polynomial PLS uses quadratic or higher order polynomial for modelling of the inner relation between ***X*** and ***Y***:

$$\boldsymbol{u}_k = f_k(\boldsymbol{t}_k) + \boldsymbol{e}_k = c_{0,k} + c_{1,k}\boldsymbol{t}_k + c_{2,k}\boldsymbol{t}_k^2 + \boldsymbol{e}_k \tag{40}$$

These kinds of nonlinear modifications to PLS require changes to the NIPALS algorithm in Equations (21)-(28). The latent variable weights ***w*** are calculated differently in the modified NIPALS algorithm as described in [21] and these types of nonlinear mappings for the PLS inner relation have been shown to be very effective at modelling nonlinear response variables. [16]

Hammerstein-Wiener models are another commonly used approach to handle nonlinearities in a measured system. The Hammerstein part of the model applies nonlinear preprocessing to the data before it is regressed by the PLS model. Finally the Wiener part post processes the linear model produced estimates. For example, a second degree polynomial can be used as the Wiener model's nonlinear part if the response variable is observed to have a second degree nonlinearity. Figure 4 shows the block diagram of the Hammerstein-Wiener model. [22]

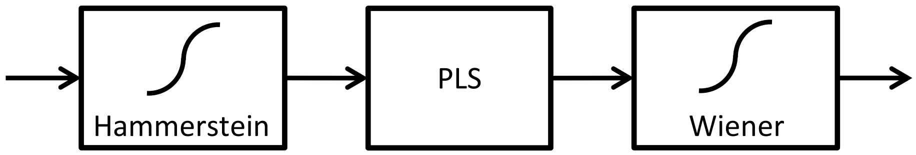


Figure 4. Hammerstein-Wiener model [22]

### 3.2.5 Priority PLS

Before using the priority PLS method, data is divided into blocks and they are arranged into a certain order. The order of the blocks dictates their importance to the modelling. The NIPALS algorithm, described in Equations (21)-(28), is used with the only difference on how the PLS weight vector is determined for each block. The algorithm is applied to each block sequentially and the weight vector values that do not belong to the current block are replaced by zeroes after Equation (21). This way the obtained score vectors refer only to the current block. The blocks are taken as parts of the model until the additional blocks do not improve performance of the model. The process data blocks can be divided according to process areas so that the further the block is from the measurement point, the lower the priority is, however, it is not always clear how the blocks should be divided and prioritized. For example, data might be divided according to spectral instruments, though in the case of multi-instrument systems it is not clear which spectral instrument should have the highest importance. Testing the model performance with different block orders is usually beneficial for finding the correct order of the blocks although with a high number of blocks this becomes a tedious task. [11]

It is highly important to estimate and correct the possible time delays when using process data for estimating properties of the final target quantity as the response between changes in the upstream and the estimated property might have a significant delay. Performance of the final multivariate model will be poor if these time delays are not taken into account. Process variable time delays can be estimated by inspecting the cross correlation functions (XCF) constructed between each of the process variables and the estimated property pairs. A sample based cross correlation at lag *t* for two data vectors is obtained from Equation (41). [23]

$$(\boldsymbol{x} \star \boldsymbol{y})[t] = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=1}^{N-t} \dfrac{(x_n - \bar{x})(y_{n+t} - \bar{y})}{\sigma_{\boldsymbol{x}}\sigma_{\boldsymbol{y}}}, & t \geq 0 \\ \dfrac{1}{N}\displaystyle\sum_{n=1}^{N+t} \dfrac{(x_{n-t} - \bar{x})(y_n - \bar{y})}{\sigma_{\boldsymbol{x}}\sigma_{\boldsymbol{y}}}, & t < 0 \end{cases} \tag{41}$$

where *t* is an integer sample lag

### 3.2.6 Multi-block PLS

Multi-block PLS handles each data block simultaneously, which is in contrast to priority PLS where the blocks are treated one at the time. The MB-PLS method gives an equal importance to each of the variable blocks and is typically used in cases where the data is generated from different types of instruments. For example, spectroscopic measurement data might be assigned to one block and other relevant process measurements to another block. [11] The blocks should be chosen so that they are not strongly related to each other as the estimation performance of a model may be poor, if the variables of different blocks are highly related. Correlation within the blocks does not affect the estimation power of the model. The following illustrates a modified NIPALS algorithm for MB-PLS: [24]

A. Preprocess each $\boldsymbol{X}$-block and $\boldsymbol{Y}$. Set $\boldsymbol{X}_1 = \boldsymbol{X}$, $\boldsymbol{Y}_1 = \boldsymbol{Y}$ and $k = 1$. Divide the data matrix into blocks $\boldsymbol{X}_{b,k}$:

$$\boldsymbol{X}_k = [\boldsymbol{X}_{1,k}, \quad \dots, \quad \boldsymbol{X}_{B,k}] \tag{42}$$

where *B* is the number of blocks

B. Choose an initial score vector $\boldsymbol{u}_k$ from one of the $\boldsymbol{Y}$ columns

C. Calculate the latent variable *k* weights for each block $\boldsymbol{X}_{b,k}$ using Equation (43).

$$\boldsymbol{w}_{b,k} = \frac{\boldsymbol{X}_{b,k}^T \boldsymbol{u}_k}{\left\| \boldsymbol{X}_{b,k}^T \boldsymbol{u}_k \right\|} \tag{43}$$

D. Calculate the latent variable *k* score vectors for each block $\boldsymbol{X}_{b,k}$:

$$\boldsymbol{t}_{b,k} = \boldsymbol{X}_{b,k} \boldsymbol{w}_{b,k} \tag{44}$$

E. Combine the block score vectors $\boldsymbol{t}_{b,k}$ into a matrix $\boldsymbol{T}_k$:

$$\boldsymbol{T}_k = [\boldsymbol{t}_{1,k}, \quad \dots, \quad \boldsymbol{t}_{B,k}] \tag{45}$$

F. Calculate a super weight for the latent variable *k*:

$$\boldsymbol{w}_{S,k} = \frac{\boldsymbol{T}_k \boldsymbol{u}_k}{\|\boldsymbol{T}_k \boldsymbol{u}_k\|} \tag{46}$$

G. Calculate a super score for the latent variable *k*:

$$\boldsymbol{t}_{S,k} = \boldsymbol{T}_k \boldsymbol{w}_{S,k} \tag{47}$$

H. Calculate the latent variable *k* loading vector:

$$\boldsymbol{q}_k = \frac{\boldsymbol{Y}_k^T \boldsymbol{t}_{S,k}}{\boldsymbol{t}_{S,k}^T \boldsymbol{t}_{S,k}} \tag{48}$$

I. Update the response variables score vector for the latent variable *k*:

$$\boldsymbol{u}_k = \frac{\boldsymbol{Y}_k \boldsymbol{q}_k}{\boldsymbol{q}_k^T \boldsymbol{q}_k} \tag{49}$$

J. Test the ***u*** score vector for convergence using Equation (50). Go to the step J if ***u*** has converged, otherwise return to the step C.

$$\frac{\|\boldsymbol{u}_{k-1} - \boldsymbol{u}_k\|}{\|\boldsymbol{u}_k\|} < \varepsilon \tag{50}$$

where $\varepsilon$ is a small convergence tolerance

K. Calculate the block loadings for the latent variable k:

$$\boldsymbol{p}_{b,k} = \frac{\boldsymbol{X}_{b,k}^T \boldsymbol{t}_{b,k}}{\boldsymbol{t}_{b,k}^T \boldsymbol{t}_{b,k}} \tag{51}$$

L. Calculate the super loadings for the latent variable k:

$$\boldsymbol{p}_{S,k} = \frac{\boldsymbol{X}_k^T \boldsymbol{t}_{S,k}}{\boldsymbol{t}_{S,k}^T \boldsymbol{t}_{S,k}} \tag{52}$$

M. Deflate the explanatory variable matrix ***X*** and the response variable matrix ***Y*** using Equations (53) and (54).

$$\boldsymbol{X}_{b,k+1} = \boldsymbol{X}_{b,k} - \boldsymbol{t}_{b,k} \boldsymbol{p}_{b,k}^T \tag{53}$$

$$\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k - \boldsymbol{t}_{S,k} \boldsymbol{q}_k^T \tag{54}$$

N. Stop if all latent variables have been calculated. Otherwise let $k = k + 1$ and return to the step B

At each step the block weights $\boldsymbol{w}_{b,k}$, block scores $\boldsymbol{t}_{b,k}$, block loading $\boldsymbol{p}_{b,k}$, super weights $\boldsymbol{w}_{S,k}$, super scores $\boldsymbol{t}_{S,k}$, response loadings $\boldsymbol{q}_k$, response scores $\boldsymbol{u}_k$ and super loadings $\boldsymbol{p}_{S,k}$ are collected into the weight, score and loading matrices $\boldsymbol{W}_b$, $\boldsymbol{T}_b$, $\boldsymbol{P}_b$, $\boldsymbol{W}_S$, $\boldsymbol{T}_S$, $\boldsymbol{Q}$, $\boldsymbol{U}$, $\boldsymbol{P}_S$ accordingly.

The MB-PLS NIPALS explanatory matrix can be deflated using two different methods. The first one is called the block score update method and it is shown in Equation (53). This method creates orthogonal block scores, but the super scores are correlated. The second deflation method is called the super score update method. This uses the super score vectors for ***X*** deflation and the following equation outlines the super score update equation:

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - \boldsymbol{t}_{S,k} \boldsymbol{p}_{S,k}^T \tag{55}$$

The super score deflation method creates orthogonal super scores, but block scores are correlated. Estimation performance of the model is worse with the block score update method, if the blocks are correlated when compared to the super score update. On the other hand, explained variance within the blocks becomes higher when using the block score update method. [25]

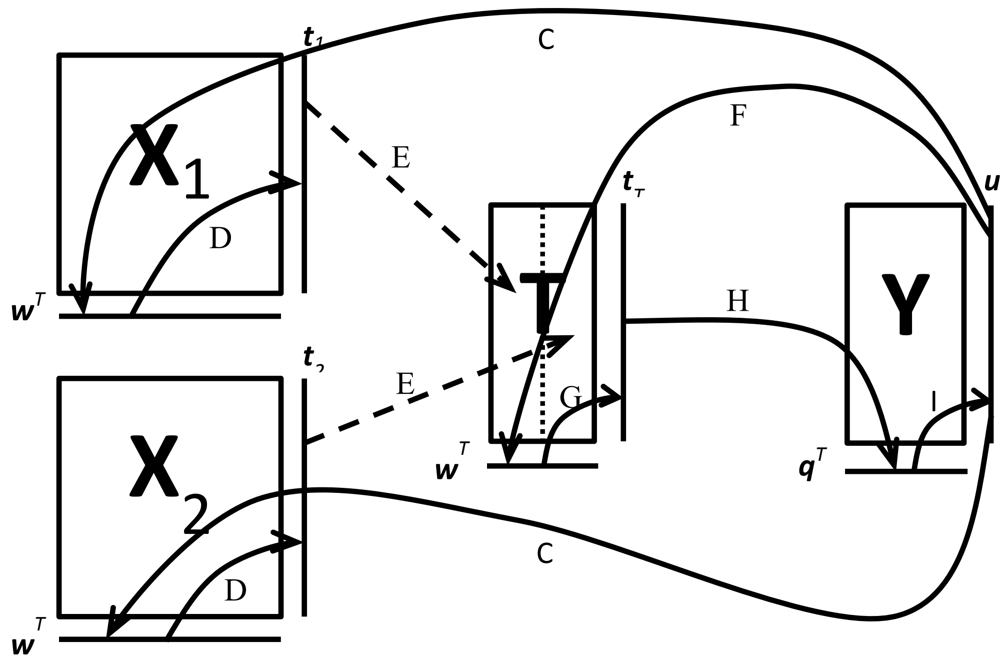


Figure 5. Iteration steps of the MB-PLS NIPALS algorithm. [24]

Figure 5. Arrow scheme diagram illustrates a single iteration step of the MB-PLS NIPALS algorithm for two data blocks. The steps C-I are repeated until the convergence of ***u***. After ***u*** has converged a super loading vector $\boldsymbol{p}_T$ is obtained in the step L and the ***X*** and ***Y*** matrices are deflated. This is repeated until all latent variables are found.

The MB-PLS regression differs from the ordinary PLS regression. The MB-PLS regression is an iterative algorithm where the data sample is first compressed into a super score vector and then into a final estimation by applying the super score to the response loading vector. The regression algorithm is outlined below: [26]

A. Preprocess a new sample $\boldsymbol{x}^{new}$ similarly as the model training data. Set $\boldsymbol{x}_1^{new} = \boldsymbol{x}^{new}$ and *k = 1*. Divide the sample vector into blocks:

$$\boldsymbol{x}_k^{new} = [\boldsymbol{x}_{1,k}^{new} \quad \boldsymbol{x}_{2,k}^{new} \quad \dots \quad \boldsymbol{x}_{B,k}^{new}] \tag{56}$$

B. Calculate a block score value for each block for the latent variable *k*:

$$t_{b,k}^{new} = \boldsymbol{w}_{b,k}^T \boldsymbol{x}_{b,k}^{new} \tag{57}$$

C. Calculate a super score value for the latent variable *k* using Equation (58).

$$t_{S,k}^{new} = [t_{1,k}^{new} \quad t_{2,k}^{new} \quad \dots \quad t_{B,k}^{new}]\boldsymbol{w}_{S,k} \tag{58}$$

D. Deflate the sample vector by removing the variance associated with the latent variable $k$. Equation (59) is used if the MB-PLS model was created using the block score update method. Otherwise Equation (60) is used.

$$\boldsymbol{x}_{b,k+1}^{new} = \boldsymbol{x}_{b,k}^{new} - t_{b,k}^{new}\boldsymbol{p}_{b,k} \tag{59}$$

$$\boldsymbol{x}_{k+1}^{new} = \boldsymbol{x}_{k}^{new} - t_{S,k}^{new}\boldsymbol{p}_{S,k} \tag{60}$$

E. Stop if all latent variables have been used. Otherwise let $k = k + 1$ and return to the step B

F. Calculate an estimate for the response variable and the block residuals. The residual Equation (62) is used if the MB-PLS model was created using the block score update method, otherwise Equation (63) is used.

$$\hat{\boldsymbol{y}}^{new} = \sum_{k=1}^{K} t_{S,k}^{new}\boldsymbol{q}_k \tag{61}$$

$$\boldsymbol{e}_b^{new} = \boldsymbol{x}_b - \sum_{k=1}^{K} t_{b,k}^{new}\boldsymbol{p}_{b,k} \tag{62}$$

$$\boldsymbol{e}_b^{new} = \boldsymbol{x}_b - \sum_{k=1}^{K} t_{S,k}^{new}\boldsymbol{p}_{S,k} \tag{63}$$

$$\boldsymbol{e}^{new} = \sum_{b=1}^{B} \boldsymbol{e}_b^{new} \tag{64}$$

It has been shown that dividing data into meaningful data blocks is beneficial for performance of the model. [11] The Multiblock PLS has the ability to increase estimation performance of a model compared to the ordinary PLS. One study combined NIR analyzer and process data to improve the estimation performance of a model [27]. Another study showed that MB-PLS applied to NIR and MIR data blocks had better performance characteristics with a lower number of latent variables when compared to the ordinary PLS [28].

$T^2$ and *SPE* statistics can be calculated separately for each of the MB-PLS blocks using the latent variables obtained for each block. [29] Statistics for the super score matrix can also be obtained. Contributions of each individual variable and block can be obtained separately. [26] Block division improves interpretation of the statistics since different data sources can be monitored separately, such that instrumental problems with a spectroscopic device can be detected separately from other measurements.

Kohonen *et al*. compared ordinary PLS, priority PLS and multiblock PLS for monitoring the fertilizer flowability and they demonstrated that a P-PLS model gave the most accurate estimates [30]. In their study the MB-PLS model required a lower number of latent variables than P-PLS while still providing a better performance than the ordinary PLS model. They also argued that MB-PLS was superior for quality monitoring of the fertilizer product since this type of model produces more information when compared to the ordinary and priority PLS models. Moreover control limits and statistics can be calculated for each of the data blocks separately in the MB-PLS model. In addition, Laxalde *et al*. have shown the successful application of the MB-PLS model for estimation of resin content in heavy oils by combining absorbance data from the NIR and MIR instruments [31]. The estimation performance of the model was shown to be improved when data was combined from multiple sources. Jing *et al.* used MB-PLS in a blockwise data analysis in order to determine the significant contributors to the quality of a pharmaceutical product produced by a multistep batch process [32]. They utilized the MB-PLS analysis results to reduce variability of the final product.

### 3.2.7 Adaptive chemometric models

It has been suggested by Angelov *et al*. [33] that variability in the operational regions of processes make utilization of the process measurements difficult for chemometric analysis. Generally it is not feasible to cover all the possible operational regions of the process during the training of the model and process dynamics also tend to change over longer periods of operation, which can cause the model to become unusable over time and require expensive remodeling of the system. Therefore adaptive chemometric models should be used in order to adapt to changes in the process. Figure 6 illustrates the general adaptive scheme used for chemometric models and also how some of the adaptive methods may directly utilize historical input data when the model is updated.

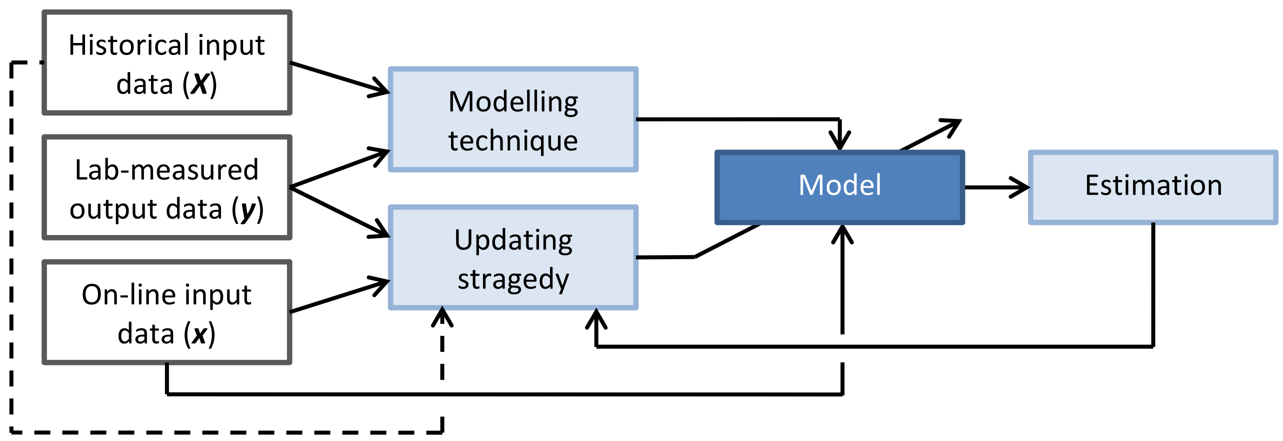

Historical input data (***X***)
Lab-measured output data (***y***)
On-line input data (***x***)
Modelling technique
Updating stragedy
Model
Estimation

Figure 6. An adaptive chemometric model [34] [35]

A variety of different types of recursive PLS methods are detailed within the literature. Qin presented a recursive PLS method where a model is updated when a new sample becomes available or after a certain number of samples have arrived [34]. The proposed model used a forgetting factor that gives less weight to older samples and the method was applied to a dataset from a catalytic reformer to estimate the product octane number. The results from this investigation showed that the recursive model successfully adapt to the changes in the process.

Updating a PCA or PLS model using a recursive method is very expensive computationally when a full data matrix is used for generating the model [36]. Therefore it has been suggested that the IKPLS algorithm, shown in Equations (31)-(38), should be used in a recursive manner instead. The method relies on a PLS model that is derived from a covariance matrix instead of full data matrices. [35]

The recursive methods with a constant forgetting factor tend to have problems with systems that are not activated sufficiently as this causes the recursive method to adapt to measurement noise, leading to a decrease in model performance. For example a covariance matrix might become ill-conditioned if used in methods that rely on recursive matrix updates. Dayal *et al*. proposed a method where the forgetting factor is also adaptive. Here the forgetting factor was updated based on the measures of information in the new and old data points with the proposed method yielding an improved estimation performance in a mining process. [37]

The previously proposed recursive PLS methods are based on adaption by prioritization of the most recent samples and such a type of adaptive scheme is also termed time relevance adaptation. Chen *et* *al*. proposed a method where not only time relevance is taken into account, but also space relevance. This method uses the locally weighted PLS method where time relevance weighting is applied to the most recent samples and space relevance weighting to the old samples with the space relevance determined from the Euclidean distance between a new sample and the historical samples. This method has been successfully used with NIR spectrum data and it yielded improved performance when compared to the other weighting adaptive schemes. [35]

A novel method was presented by Angelov *et al*. that used the Evolving Takagi-Suveno Fuzzy Model for adaptive modelling of a distillation column dataset [33]. This type of a model is based on fuzzy logic where rules as well as parameters of the model evolve [38]. Use of this model yielded improved performance over other adaptive chemometric schemes in a variety of test cases [38] [39].

However, all adaptive models have the inherit problem that they may adapt to unwanted operational process states like process disturbances and shutdowns. For example a model might become unusable after it has adapted to a dataset from a process that has been shutdown. Therefore it might be necessary to create certain conditions for model updating such as to make adaptive models feasible for the real process applications. These conditions could be based on prior knowledge of the system. [40]

## 3.3 Assessment of the model complexity

The selection of the PCA principal components can be based on the variation explained by PCs in the data. The optimal number of principal components (*K*) can be estimated by the following criteria:

$$\widehat{K} = \arg\min_{K} \left\{ K : \sum_{k=1}^{K} \lambda_k \geq (1-\varepsilon) \sum_{k=1}^{L} \lambda_k \right\} \tag{65}$$

where $\lambda_k$ is the *k*:th eigenvalue of the ***X*** covariance matrix which is square of a singular value in a column of ***S***

*K* is a number of the selected principal components

*L* is a total number of the principal components

*ε* is a fraction of the unexplained variance

Selection of the parameter ε value greatly depends on the amount of noise in the data. If the level of the noise is not known in advance the selection of this parameter can be uncertain, however in such cases the fraction of unexplained variance is selected to be close to 10%. [18]

A number of studies have shown that the statistical F-test can be employed with noisy data to evaluate the number of principal components [18] [41]. An F-test starts from the smallest eigenvalue and approaches the largest one until it finds a significant eigenvalue and once this has been determined all larger eigenvalues are also considered to be significant. The F-statistic is calculated for each eigenvalue (*k*) using the following formula:

$$F_k = \frac{\dfrac{\lambda_k}{(M-k+1)(L-k+1)}}{\sum_{i=k+1}^{L} \dfrac{\lambda_i}{(M-i+1)(L-i+1)}} \tag{66}$$

where *M* is the number of variables

Significance of a principal component is tested by comparing the above test statistic against a one-tailed critical value of F-distribution at a desired level of confidence. The estimated number of principal components can be determined using the following relation:

$$\widehat{K} = \arg\min_{K}\{K: F_K > f_{1,L-K}(\alpha)\} \tag{67}$$

where $f_{1,L-K}(\alpha)$ is F-distribution's critical value at $\alpha$ with 1 and *L-K* degrees of freedom

The choice of a significance level α is again under users' discretion, as was the case with the parameter ε in Equation (65), as a consequence the level of significance is usually set between 90% and 95%. [17]

Another common approach to estimate the correct order of a PCA model is to use cross-validation techniques that divide a dataset into training and validation datasets. A model is built using the training dataset and validated using the validation set. Validation is performed by comparing the estimated values obtained from the model to the ones in the validation dataset.

The prediction error sum of squares (*PRESS*) is commonly used to interpret the performance of the model as shown in Equation (68).

$$PRESS = \sum_{n=1}^{N} \sum_{m=1}^{M} (\hat{x}_{nm} - x_{nm})^2 \tag{68}$$

This performance indicator is usually normalized, and the resulting root mean squared error of cross-validation (*RMSECV*) is given by: [11]

$$RMSECV = \sqrt{\frac{PRESS}{N}} \tag{69}$$

Performance of the model fitting can also be measured using the coefficient of determination ($R^2$) which indicates how well the estimated values fit to the observed data set. Usefulness of the $R^2$-metric is limited when data contains outliers or non-linear behavior though this can be evaluated by visual inspection of the estimated values plotted against the observed ones. Such an evaluation allows for the detection of possible outliers in the dataset and non-linear behavior of the system. The following equation shows the formulation for the coefficient of determination:

$$R_m^2 = 1 - \frac{\sum_{n=1}^{N} (\hat{x}_{nm} - x_{nm})^2}{\sum_{n=1}^{N} (x_{nm} - \bar{x}_m)^2} \tag{70}$$

The impact of successive latent variables can be assessed using the $R_R$ criterion. The $R_R$ metric measures how influential an added latent variable is to the performance of the model. Usually a value of 0.90-0.95 is used as a cut-off limit for the criterion and latent variables exceeding this limit are regarded as redundant. The $R_R$ metric is calculated using Equation (71). [11]

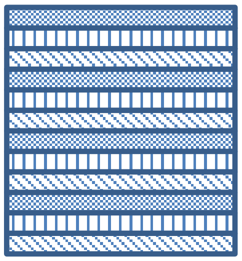
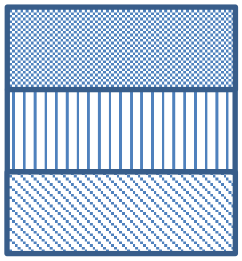
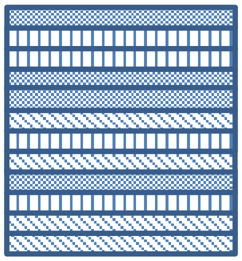
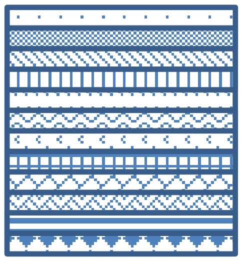
$$R_R = \frac{PRESS_{k+1}}{RSS_k} \tag{71}$$

where $k$ is an index of the latent variable

*RSS* is a residual sum of squares $RSS = (\boldsymbol{x}^T \boldsymbol{x}) - (\hat{\boldsymbol{x}}^T \hat{\boldsymbol{x}})$

Many cross-validation technique variants have been introduced throughout the literature and this thesis outlines the following: Venetian Blind, Contiguous Block, Monte-Carlo and Leave-one-out cross validation techniques. All these methods divide a dataset into two groups with the first group used to train the model and the second one is used for the model validation. The Venetian Blind, Contiguous Block and Monte-Carlo cross-validation methods divide the dataset into $N_v$ parts. The Venetian Blind method divides the dataset by taking every $N_v$:th datum from the dataset, the Contiguous Block method divides the dataset into contiguous parts and the Monte-Carlo cross-validation divides the dataset into $N_v$ parts by randomly choosing the samples from the dataset. The Monte-Carlo method is repeated $N_{rep}$ times by taking different randomly chosen data subsets, whilst the Leave-one-out method excludes a single sample from the dataset and the rest are used for model training. The model is then used to estimate the left-out sample. A table below illustrates these four commonly used cross-validation methods. [42]

Table 1. Cross-validation schemes

| | Venetian Blinds | Contiguous Blocks | Monte-Carlo | Leave-one-out |
|---|---|---|---|---|
| Sample selection scheme |  |  |  |  |
| Number of validation experiments | $N_v$ | $N_v$ | $N_v * N_{rep}$ | $N$ |
| Number of validation samples per experiment | $N/N_v$ | $N/N_v$ | $N/N_v$ | 1 |

*RMSECV* values are calculated for each of the validation experiments and these values are then summed together to obtain the total *RMSECV* value. This procedure is repeated for different model orders, then the total *RMSECV* values are compared and the correct model order selected at the point where model performance is not improved by successive latent variables. It has been suggested that the optimal number of principal components determined by the Monte-Carlo method is based on lowest value of the total sum of the *MCCV* metric rather than *RMSECV*. [11]

$$MCCV = \frac{1}{N_{rep}N_v}\sum_{n=1}^{N}\sum_{m=1}^{M}(\hat{x}_{nm} - x_{nm})^2 \tag{72}$$

where $N_{rep}$ is a number of *MCCV* repetitions and $N_v$ is a number of validation samples

The appropriate cross-validation technique should be selected based on structure of the dataset, for example, the Venetian Blind method is good for randomly ordered datasets that do not contain replicates. On the other hand the Contiguous Block method is useful for evaluating model stability for datasets containing batch data whilst the Monte-Carlo cross-validation is used when the structure of the dataset is not known or when the dataset contains a low number of samples. Monte-Carlo is also a very versatile method because it can be applied to many different types of datasets, however, it is computationally very intensive due to the large number of repetitions required. In contrast, the Leave-one-out method is usually only used for very small datasets.

Information about response variables can also be used to determine the correct model order for PCR and PLS models. Explanatory and response datasets are both divided into training and validation datasets with the model validated against the validation set of the response variables. The Root mean squared error of the prediction (*RMSEP*) performance metric is calculated for different model orders for the validation data. The *RMSEP* is calculated using Equation (69) where the response variable is used instead of explanatory variable sample as in the *RMSECV*. Response variables are similarly used in the Monte Carlo cross-validation Equation (72). The model producing the optimal *RMSEP* or *MCCV* performance is determined to have the correct number of principal components or latent variables. [11]

It has been suggested that the optimal selection of the latent variables in MB-PLS is based on the inequality shown is Equation (73).

$$\left(\boldsymbol{t}_{b,k}^{T}\boldsymbol{u}_{k}\right)^{2} > \frac{1+\sqrt{2}\boldsymbol{\Phi}^{-1}(\alpha)}{N}\left(\boldsymbol{t}_{b,k}^{T}\boldsymbol{t}_{b,k}\right)\left(\boldsymbol{u}_{k}^{T}\boldsymbol{u}_{k}\right) \tag{73}$$

where $\boldsymbol{\Phi}^{-1}(\alpha)$ is a value of a probit function with a specific probability $\alpha$. The probit function is an inverse of the normal distributions cumulative distribution.

The significance of a latent variable is tested using this inequality and a significance level of $\alpha = 0.95$ is usually used for determining significant scores. The testing of latent variables is stopped after no significant latent variable is found for any of the blocks. Blocks may have a different number of latent variables since insignificant latent variables are discarded from the MB-PLS model and an unequal number of latent variables between the blocks help to avoid overfitting within a block. The relation outlined in Equation (73) can also be utilized for ordinary PLS model order selection. [11] The MB-PLS weight, score and loading vectors must be re-evaluated using the MB-PLS NIPALS algorithm when any of the blocks have an unequal number of latent variables after the model order selection. Insignificant latent variables are then excluded in the MB-PLS NIPALS Equations (43)-(47).

## 3.4 Selection of the significant regressors

Data produced by spectrometric instruments contain usually hundreds to thousands of variables; therefore it is important to select only the relevant ones for modelling. Removing irrelevant variables is advantageous since the performance of the model is then only affected by the presence of relevant variables and computational performance can also increase due to the lower number of variables. PLS regression can be utilized for determining influential explanatory variables. One commonly used method is Variable Importance in Projection (*VIP*) method that calculates a value signifying the importance of the variable. For example, a variable whose *VIP* value is close or above one is considered a significant variable and it has been suggested that the significance threshold is between 0.83 and 1.21. The *VIP* value can be calculated for each variable (*m*) using the following equation: [43]

$$VIP_m = \sqrt{M \sum_{k=1}^{K} \left[ (\boldsymbol{q}_k^T \boldsymbol{q}_k \boldsymbol{t}_k^T \boldsymbol{t}_k) \left( \frac{w_{km}}{\|\boldsymbol{w}_k\|} \right)^2 \right] \Big/ \sum_{k=1}^{K} (\boldsymbol{q}_k^T \boldsymbol{q}_k \boldsymbol{t}_k^T \boldsymbol{t}_k)} \tag{74}$$

where $w_{km}$ is a weight corresponding to the latent variable k for the variable m

The *VIP* significance is calculated using variance and sum of squares explained by the PLS component. Variance explained is calculated in the term $(w_{km}/\|\boldsymbol{w}_k\|)^2$, whilst the term $(\boldsymbol{q}_k^T \boldsymbol{q}_k \boldsymbol{t}_k^T \boldsymbol{t}_k)$ represents the sum of squares explained by the latent variable. Thus the *VIP* value therefore represents each variable's relative importance on the model.

Another commonly used method for assessing variable importance is one that uses regression coefficients in a regression matrix ***B***. For example, variables whose regression coefficient is below a certain limit are regarded as irrelevant and removed. The Uninformative Variable Elimination (UVE) algorithm derives a variable importance metric from regression coefficients. The algorithm appends noise variables to the data matrix and forms a variable exclusion limit from these appended variables and is outlined next. [44]

A. Set $\overline{PRESS}_0 = \infty$ and $i = 1$

B. Append the explanatory matrix ***X*** with *M* random variables drawn from a uniform distribution

C. Determine the mean values and variances of each regression coefficient from the different PLS models derived by the Monte-Carlo cross-validations. Also calculate the average $\overline{PRESS}_i$ of the *PRESS* values for the Monte-Carlo derived models.

D. Stop if the average of the *PRESS* values starts to increase, otherwise go to the step E:

$$\overline{PRESS}_i > \overline{PRESS}_{i-1} \tag{75}$$

E. Calculate an importance metric for each variable by using means and variances of the regression coefficients:

$$c_m = \frac{\bar{b}_m}{\sigma^2_{\boldsymbol{b}_m}} \tag{76}$$

where $\bar{b}_m$ is the mean value of the variable *m* regression coefficients

$\sigma^2_{\boldsymbol{b}_m}$ is the variance of the variable *m* regression coefficients

F. Calculate a variable exclusion limit by finding the maximum value of the importance metrics $c_m$ belonging to the random variables appended in the step A.

G. Remove the variables whose importance $c_m$ is below the exclusion limit. Stop if no variables are removed otherwise return to the step B.

The *VIP* value obtained in Equation (74) can be used to select the relevant variables. It can be also used in an iterative variable elimination method called the Backward Variable Elimination (BVE) which is outlined below: [45]

A. Calculate the mean values of the *PRESS* and the variable *VIPs* for the different PLS models obtained by the Monte-Carlo cross-validations.

B. Eliminate the variable corresponding to the smallest *VIP* value

C. Go to the step D if the number of variables is the same as the number of the latent variables after elimination, otherwise return to the step A.

D. Select excluded variables based on the smallest *PRESS* value obtained in the elimination iterations

It is worth noting that the BVE algorithm can also be based on another importance metric e.g. regression coefficients can be used in place of the *VIP* values. This UVE algorithm is well suited for datasets containing a large number of variables like spectroscopic measurements as each iteration step of the UVE algorithm removes multiple, uninformative variables. In contrast, the BVE algorithm is computationally very expensive for datasets containing a large number of variables since it removes only one variable per iteration step. [45]

Both of these elimination algorithms work backwards by removing variables step by step. Covariance procedure (COVPROC) is a forward variable selection method where variables are included to the model step by step based on the latent variable weight $w_{km}$. Thus a variable whose weight is the highest is included to the model and variables are included into the model until a model fitting criteria, for example, the *RMSEP* value no longer improves. [27] [43] Mehmood *et. al.* have reviewed numerous other elimination methods for the PLS method [43] and their findings detail how some of them rely on the stochastic method such as genetic algorithms whilst other rely on a more complex iterative schemes.

## 3.5 Multivariate statistical process control

Statistical process control (SPC) methods are used to monitor the status of a process with faults in the process and quality of the measurements which are the usual targets of the monitoring. The Hotelling's $T^2$ and squared prediction error (*SPE*) statistics are the most widely used multivariate model performance indicators. These statistics are usually used to monitor the performance of the PCA or PLS models as well as the operational performance of the process. The $T^2$-statistic measures the samples' unexplained variation inside the model and it is usually used to indicate whether a model fit is good. The *SPE*-statistic monitors the model residuals and is used to measure unexplained variation outside the model in order to indicate any possible unexplained events in the process. The *SPE* and $T^2$-statistics can be thus used to indicate process faults and the presence of unusual operational regions within the process. [46]

The $T^2$ and *SPE* statistics can be applied to PCA and PLS models. Statistics can be calculated in a similar fashion for both types of models since their underlying mathematical framework is closely related. The $T^2$ statistic of a measurement *n* from a sample can be calculated using the following equation: [11]

$$T_n^2 = \sum_{k=1}^{K} \frac{t_{nk}^2}{s_{\boldsymbol{t}_k}^2} \tag{77}$$

where *K* is the number of the latent variables used for the model

$t_{nk}$ is the *n*:th row and the *k*:th column of the PCA or PLS score matrix $\boldsymbol{T}$ obtained from a new sample

$s_{\boldsymbol{t}_k}$ is the estimated variance of the modelling phase score vector $\boldsymbol{t}_k$

The estimated variances are obtained from the covariance matrix diagonal elements of the score matrix $\boldsymbol{T}$ at the modeling phase. A score matrix covariance matrix is calculated using the following equation:

$$\boldsymbol{S_{TT}} = \frac{1}{N-1}\boldsymbol{T}^T\boldsymbol{T} \tag{78}$$

where $\boldsymbol{T}$ is the score matrix of the PCA or PLS model

*N* is the number of samples used for modeling

An upper control limit of the $T^2$ statistic can be calculated from the following equation: [11]

$$T_{lim}^2 = \frac{K(N-1)}{N-K} f_{K,N-K}(\alpha) \tag{79}$$

where *N* is the number of samples

$f_{K,N-K}(\alpha)$ is a critical value of the F-distribution at $\alpha$ with *K* and *N-K* degrees of freedom

The choice of the significance level α is under user's discretion and a usual value for this parameter is between 95% and 99.9%.

The SPE statistic is calculated for a measurement *n* in the sample using Equation (80). [11]

$$SPE_n = \boldsymbol{e}_n^T \boldsymbol{e}_n \tag{80}$$

where $e_n$ is the *n*:th row of the residual matrix, which is calculated using Equation (81).

$$\boldsymbol{E}_{new} = \boldsymbol{X}_{new} - \boldsymbol{T}_{new}\boldsymbol{P}^T \tag{81}$$

The upper control limit for the SPE statistic is calculated using the following equation for a PCA model [47]:

$$SPE_{lim} = \Theta_1 \left( 1 + \frac{\boldsymbol{\Phi}^{-1}(\alpha) h_0 \sqrt{2\Theta_2}}{\Theta_1} + \frac{h_0 \Theta_2 (h_0 - 1)}{\Theta_1^2} \right)^{\frac{1}{h_0}} \tag{82}$$

where $\Theta_a$ is the sum of unexplained variance associated with unused principal components

$\boldsymbol{\Phi}^{-1}(\alpha)$ is a value of the probit function with a confidence level $\alpha$. The probit function is inverse of the normal distributions cumulative distribution.

$h_0$ is calculated using the equation $h_0 = 1 - (2\Theta_1\Theta_3)/(3\Theta_2^2)$

The above $\boldsymbol{\Theta_a}$ values are calculated from variances associated with unused principal components:

$$\Theta_a = \sum_{k=K+1}^{M} s_{\boldsymbol{t}_k}^{2a} \tag{83}$$

Equation (83) requires calculation of all the latent variables related to the data and this task may be computationally very expensive for data sets with a large number of variables. Such an undertaking is especially problematic with chemometric data. Additionally the SPE limit obtained in Equation (82) is inconveniently tight for monitoring explanatory variable residuals in PLS applications (as reported by Yin *et. al.* in [48]) and this is a result of the balance between inner and outer modelling fitting in PLS. Instead, it has been suggested that the SPE control limit should be derived from the $\chi^2$-distribution: [48]

$$SPE_{lim} = g\chi_h^2(\alpha) \tag{84}$$

where *g* is a coefficient of the limit

$\chi_h^2(\alpha)$ is a critical value of the $\chi^2$-distribution at *α* with *h* degrees of freedom

Parameters *g* and *h* are calculated by using Equations in (85).

$$g = \frac{\sigma_{\boldsymbol{SPE}}^2}{2\overline{SPE}}, \quad h = \frac{2\overline{SPE}^2}{\sigma_{\boldsymbol{SPE}}^2} \tag{85}$$

where $\sigma_{\boldsymbol{SPE}}^2$ is the variance of the *SPE* values obtained in Equation (80)

$\overline{SPE}$ is the mean of the *SPE* values obtained in Equation (80)

Parameters $s_{\boldsymbol{t}_k}$ in Equation (77), $T_{lim}^2$ and $SPE_{lim}$ are obtained during the modelling phase.

The individual contributions of each variable to the statistical metrics can be calculated for both $T^2$ and *SPE* statistics. Variables with high contribution values have more influence on the observed behavior, for example, the source of a detected fault can be diagnosed by observing which variables contribute to the statistical metrics. $T^2$ contributions for each variable can be calculated by using the following equation: [11]

$$\boldsymbol{c}_{T^2,n} = \boldsymbol{t}_n \sqrt{(\boldsymbol{S_{TT}I})^{-1}} \boldsymbol{P}^T \tag{86}$$

where $\boldsymbol{I}$ is an identity matrix

$\boldsymbol{t}_n$ is a score vector of the *n*:th measurement in a new sample

*SPE* statistic contribution is simply the *n*:th measurement residual vector: [11]

$$\boldsymbol{c}_{SPE,n} = \boldsymbol{e}_n \tag{87}$$

PCA related SPC equations can be directly applied to the PLS method and the recent research papers by Yin *et. al.* and Zhou *et. al.* have also suggested modifications to the calculation of the PLS statistical values [48] [49]. The modifications are needed because the PLS method may generate large variation in explanatory variable residuals since the objective of the method is to not only minimize this variation, but also to maximize the covariance between the response variables. The papers by Yin *et. al.* and Zhou *et. al.* suggest that the PLS method should involve decomposition of data matrices into subspaces. These resulting subspaces are related to the prediction of response variables, inner relations and non-predictive parts and thus can be monitored separately.

# 4. Communication standards for the analyzer instruments

This thesis introduces a data acquisition system for spectroscopic instruments in combination with other process data sources. This chapter outlines a number of the widely used communication protocols in industrial automation particularly in analyzer applications with particular emphasis is given to the OPC UA communication protocol as the specification includes an open standard for spectroscopic instrument data acquisition.

Measurements and other data produced by spectral instruments must be transferred from the instruments to locations where the data is utilized. The analyzer results, produced in the form of spectra and response variable estimates, are used for various tasks including process control, quality control and process monitoring. Also, these results are usually saved to a database where they are available for further analysis. Milliampere Modbus, TCP/IP Modbus, Profibus and OPC (Open Platform Communications) are widely used protocols for transferring data from spectroscopic instruments, however, all analyzer manufacturers have their own software solutions to access data within devices. What data is available for external use and how it is available is highly dependent on the type of the device and manufacturer. The analyzer devices companion specification (ADI) for the OPC UA is the first standard that is intended to create a unified view on the information produced by analyzers. This thesis focuses on data transfer solutions for on-line process spectroscopic instruments. Emphasis is focused on the OPC UA (a successor to OPC) standardized by IEC (International Electrotechnical Commissions) under standard the number IEC 62541 [50].

## 4.1 OPC (Open Platform Communications)

OPC is a standardized communication model for industrial automation applications. Products compatible with the OPC specification can be used with other OPC compliant products and therefore the end users can flexibly choose between devices and manufacturers. In addition software and hardware development also becomes easier for manufacturers since no additional software solutions are required for communication between various devices. The OPC communication is based on the Distributed Component Model (DCOM) which is a proprietary technology developed by Microsoft where software components are distributed across computer networks. Figure 7 shows how client applications connect to OPC servers. OPC compliant devices operate as OPC servers and OPC clients can then connect and retrieve data from these servers. [51]

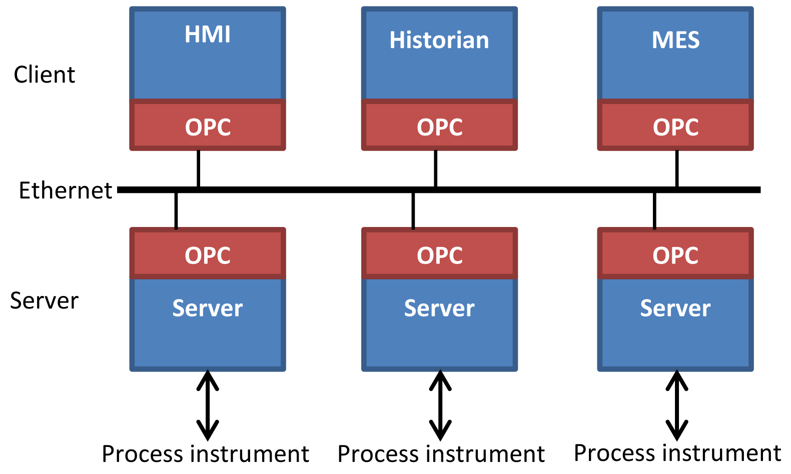


Figure 7. Communication between OPC compliant devices [51]

The OPC information model is based on groups of items. An OPC group is a logical group of items that are related to each other. An OPC item can e.g. represent a value of a measurement. Clients can request information about OPC groups and items under a specific group from the server.

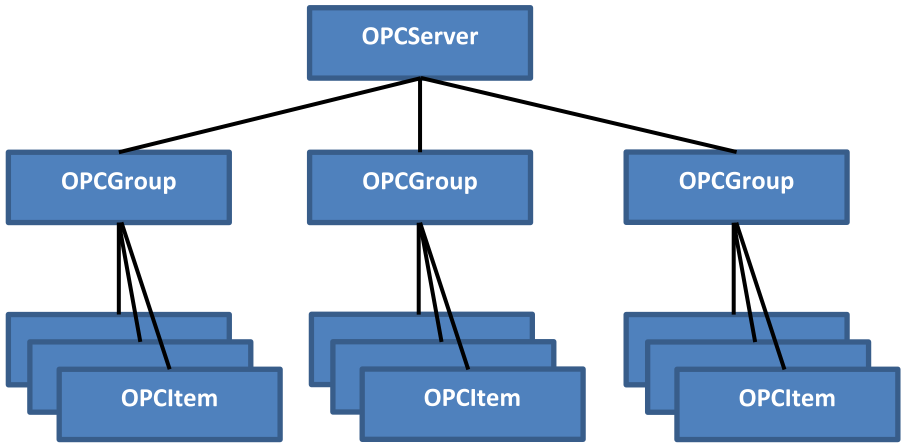


Figure 8. A hierarchy of the OPC groups and items located on an OPC server [51]

The OPC specification is divided into 11 specification documents each describing certain aspects of the standard. These include real time and historical data access specifications. The Data access specification defines how real time data is presented and accessed on a server. Alarms and events also have their own specifications. The software implementation of OPC is based on the Component Object Model (COM) developed by Microsoft and therefore OPC is not an inherently cross platform technology as each OPC object is represented by a COM object exposing specific interfaces for interacting with the component. [51]

The original OPC has no standardized information models for analyzer devices. Instead each instrument vendor has their own proprietary solutions for instrument information models. Next chapters present an OPC UA communication protocol that is the successor of the OPC. OPC UA includes a companion specification related to analyzer information models. This makes it possible to easily connect to instruments provided by different vendors.

## 4.2 OPC UA

OPC UA (Open Platform Communications Unified Architecture) is the successor of OPC and differs significantly from the original OPC as OPC UA does not rely solely on Microsoft technologies. Instead OPC UA is based on the cross-platform service-oriented architecture relying on TCP/IP communications. In particular the specification for OPC UA was created with scalability and security in mind as it supports SSL encryption for secure communication. [52]

### 4.2.1 OPC UA information models and services

The information model of OPC UA is based on full mesh node network, where the nodes are basic components of the OPC UA information model and as such can have references to other nodes thus creating relationships between them. The nodes can be thought as objects in object oriented programming as they can have child nodes representing the object properties. Furthermore, references allow the creation of full mesh networks of nodes and they can also be used to organize and create inheritance models of objects. In addition, nodes include their own metadata e.g. variable value and read access of the value. [52] This combination of nodes and references between nodes can be used to create any data model necessary.

The address space of the OPC UA server provides different types of services for obtaining information about nodes residing inside the server address space. The OPC UA specification defines *browse* and *query* services for obtaining information about the structure of a node network. The server may also contain restricted subsets of the full address space called *views*. The address space *views* are used to organize large address spaces into smaller logical parts. Figure 9 illustrates the OPC UA address space.

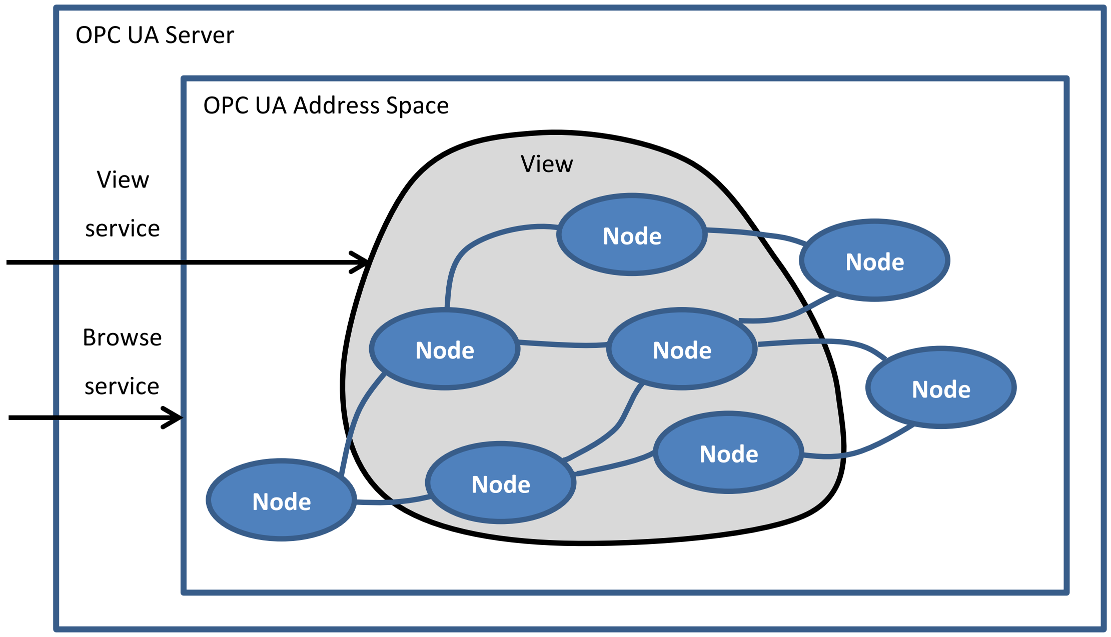


Figure 9. A mesh network of nodes inside the OPC UA server address space [51]

OPC UA nodes can be used to create information models familiar from object oriented programming. The specification defines three different kinds of nodes used for creating object instances: object, variable and method nodes. The specification also defines the node types used for information modelling, namely object type, variable type, data type and reference type nodes. These node types are referred to as node classes (*NodeClass*).

The OPC UA server provides different services based on class of the node, for example, object nodes provide services for listening to object events like alarm events. Variable nodes provide the read and write services for accessing data values as well as the services for listening to data change events. Method nodes provide services for calling methods with or without arguments and also the ability to deliver possible return values. Figure 10 shows a possible structure of an OPC UA object node with variable and method nodes. This figure also shows services provided by different instance nodes. [51]

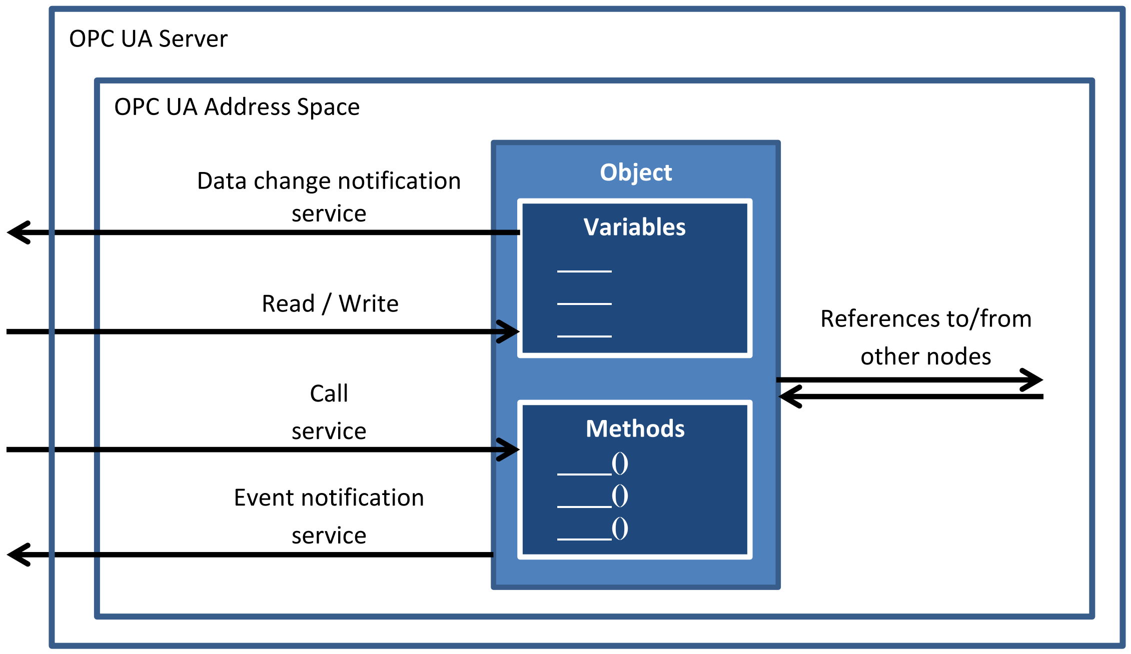


Figure 10. An individual object node inside OPC UA server address space [51]

OPC UA information models and address spaces are modelled using OPC UA notation, which consists of symbols and arrows used for denoting the nodes and references respectively. This notation is subsequently widely used throughout this thesis for information modelling. Figure 11 shows the instance node symbols on the left and the type node symbols on the right. Type nodes are used for creating type definitions that their instances refer to and this type definition hierarchy describes how an initiated type is represented inside an address space. Again, this paradigm is familiar from object oriented programming where classes define a collection of properties and methods for the objects.

A name inside a node symbol represents the *BrowseName* of the node i.e. it reflects the node's name when address space of the server is browsed. Each of the different node symbols represents an OPC UA *NodeClass* and the names of the specific *NodeClasses* are shown inside the symbols in Figure 11.

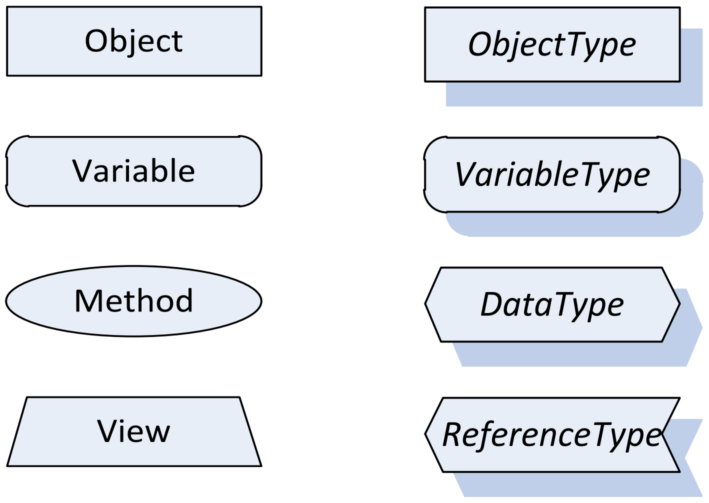


Figure 11. Node symbols in OPC UA notation [51]

The OPC UA standard defines a set of reference types - either symmetric or asymmetric - as shown in Figure 12. In addition, custom reference types can also be defined and all references are described in an address space by reference type nodes. Hierarchical references are used for creating non-circular references between nodes inside an address space. Overall, each reference type is represented by its name over the arrow symbol. A *HasEventSource* reference is used for describing a source of a possible object event. In contrast, a *HasComponent* reference can be used to describe a relation between an object and its components, for example method or variable nodes. A *HasProperty* reference is used to describe property of an object and nodes referenced by a *HasProperty* reference are regarded as variable nodes that do not have any child nodes. A *HasTypeDefinition* describes what object or variable type a node implements. In addition, a *HasSubtype* reference is used to describe an inheritance relation between object and variable types.

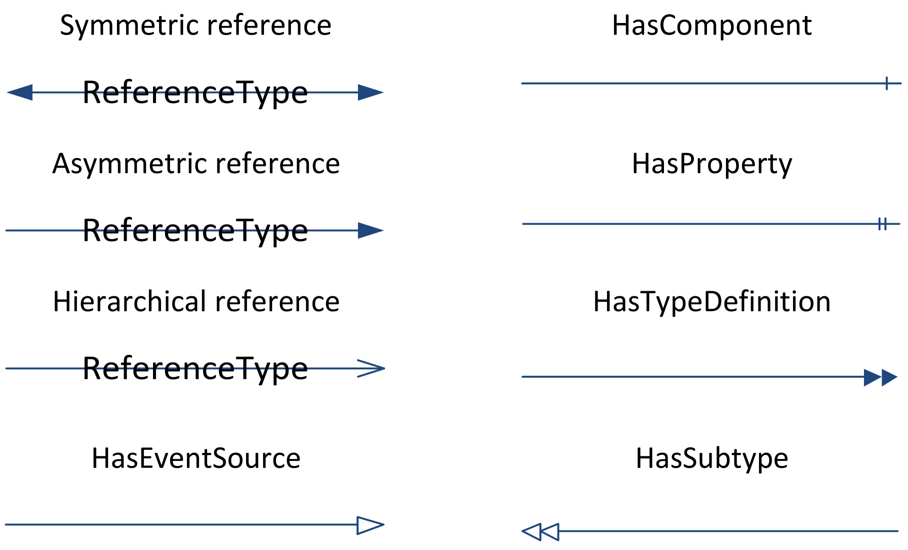


Figure 12. Reference arrow symbols in OPC UA notation [51]

*HasSubtype* and *HasTypeDefinition* relationships between nodes can also be represented inside node symbols. In addition, it is also possible to specify a *DataType* inside a variable node symbol. This notation is used to simplify OPC UA notation diagrams as outlined in Figure 13:

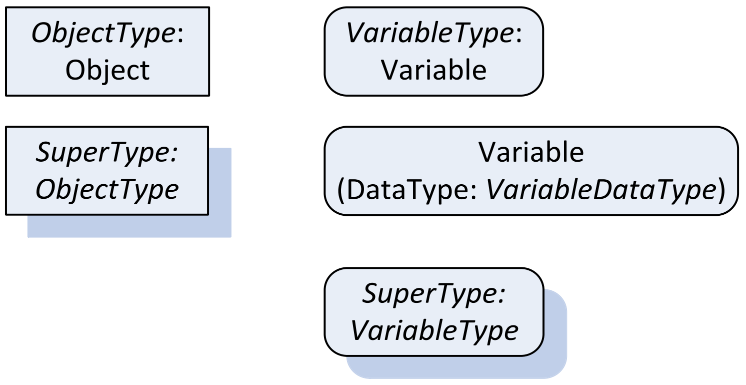


Figure 13. A notation for representing *HasSubtype, HasTypeDefinition* and *DataType* relationships inside node symbols [51]

*DataType* definitions are only defined for *Variable NodeClasses* and as such, describe what type of data is present inside a variable node. These data types include most of the familiar primitive types from common programming languages such as strings, integers and floating point types of different precisions. OPC UA also supports custom complex *DataTypes*, which are defined by subtyping the

*DataType* node. Encoding and decoding procedures for these are defined within the OPC UA server and this allows the connecting client to correctly interpret custom *DataTypes*. [51]

### 4.2.2 OPC UA ADI

The OPC UA companion specification for analyzer devices (OPC ADI) defines the information models associated with analytical devices. The model described is intended to unify the view of analytical devices irrespective of the underlying protocols. Analyzers can be divided into various groups such as light spectrometers, particle size monitoring systems, mass spectrometers, chromatograms and nuclear magnetic resonance spectrometers (though these groups can be further extended where necessary). The ADI information model can be applied to all of these analyzer groups and the specification is such that it can also support more specialized analyzers which are built by combining multiple analyzers into one package.

This thesis focuses on data acquisition from spectrometric instruments with the acquired information later used to implement a data acquisition system for analyzer devices. The configuration and maintenance specific parts in the ADI specification are disregarded.

The description of analyzers is divided into five components and these components are: *AnalyserDeviceType*, *AnalyserChannelType*, *StreamType*, *AccessoryType* and *AccessorySlotType*. Additional components can be defined depending on the needs of individual device manufacturers. Figure 14 presents a conceptual topology of the analyzer components in ADI information model. [53] Overall, *StreamType* and its subtypes are the most important types of the ADI specification for the analyzer data acquisition and it is these types that are used later in the experimental part of the thesis for collecting spectral data from the analyzer devices.

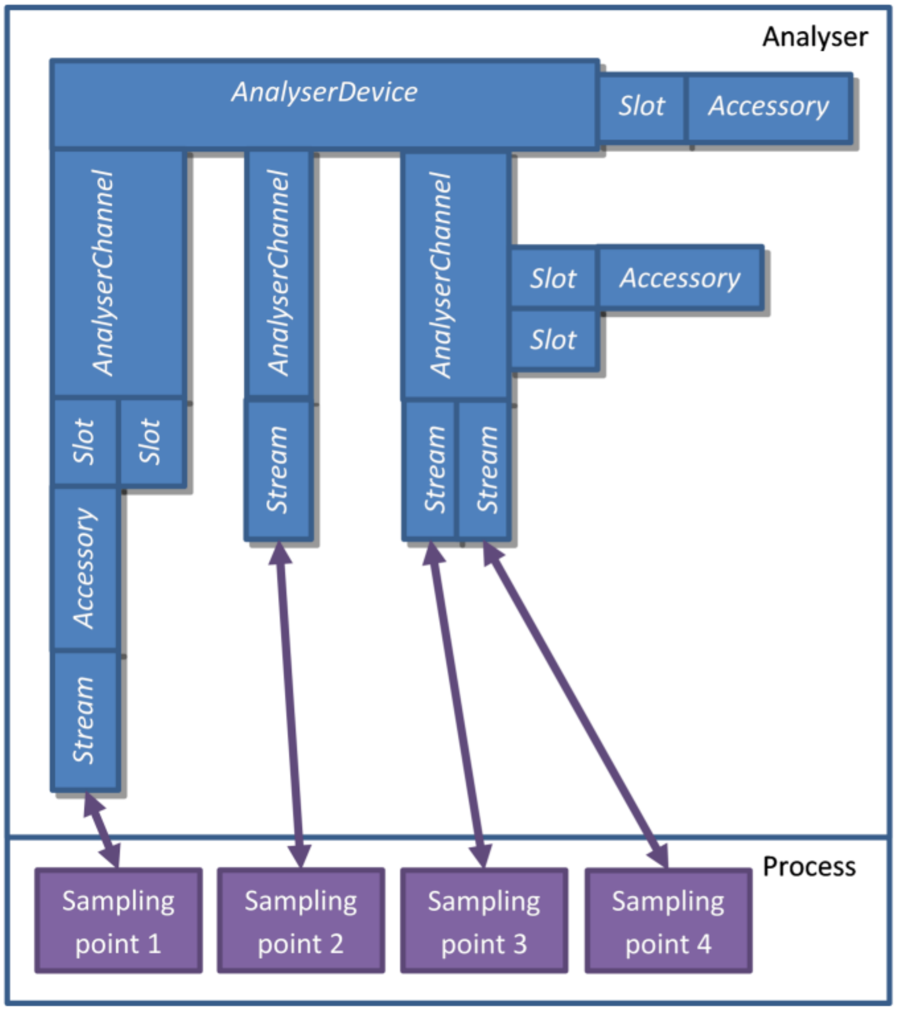


Figure 14. Analyzer components in the ADI information model [53]

*The AnalyserDeviceType* represents the instrument as a whole. A device may have multiple *AnalyserChannelType* and *AccessorySlotType* instances. Channels can also have their own *AccessorySlotType* instances. A process is sampled through a sampling system. This is modelled with the *StreamType*. Channels have one or more *StreamType* instances associated with them. *StreamType* instances may also be associated to channels through accessories. Multiple *AnalyserChannelType* instances can be active at the same time meaning that the channels are used to acquire data. Only one *StreamType* instance per channel can be active at a time.

## 4.3 Fieldbus communication standards

Fieldbus standards are widely used for transferring information inside a hierarchical automation network. This network goes all the way through from a field device to a computer screen of the operator. Fieldbus networks are usually used at the lower levels of the automation hierarchy to transfer information from a programmable logic controller (PLC) to field devices. Automation industry has a variety of competing fieldbus standards including Foundation Fieldbus, Profibus and ModBus. Analyzer devices usually support some of these fieldbus standards to transfer information from the instrument to where information is used. Although the communication protocols are standardized the underlying information models of analyzers are not. As a consequence, each instrument vendor has their own proprietary solutions on how the information is presented and in general, vendors provide their own software for connecting to the analyzer and to obtain information about the measurements. The OPC UA ADI standard is the first time when a united view of the analyzer information model has been created and this standard makes it possible to both connect to and read information from instruments without proprietary software.

## 4.4 Implementation of the standards on analyzer networks

Analyzer outputs i.e. the sample property estimates are usually transferred from the instrument to a plant distributed control system (DCS) via Ethernet or milliampere cabling. Many different types of analyzer networks might exist in parallel especially in large industrial applications. For example some of the older instruments might use Modbus milliampere messages, while other devices use OPC through Ethernet cabling [54] [55]. Figure 15 depicts an example analyzer network in this type of configuration:

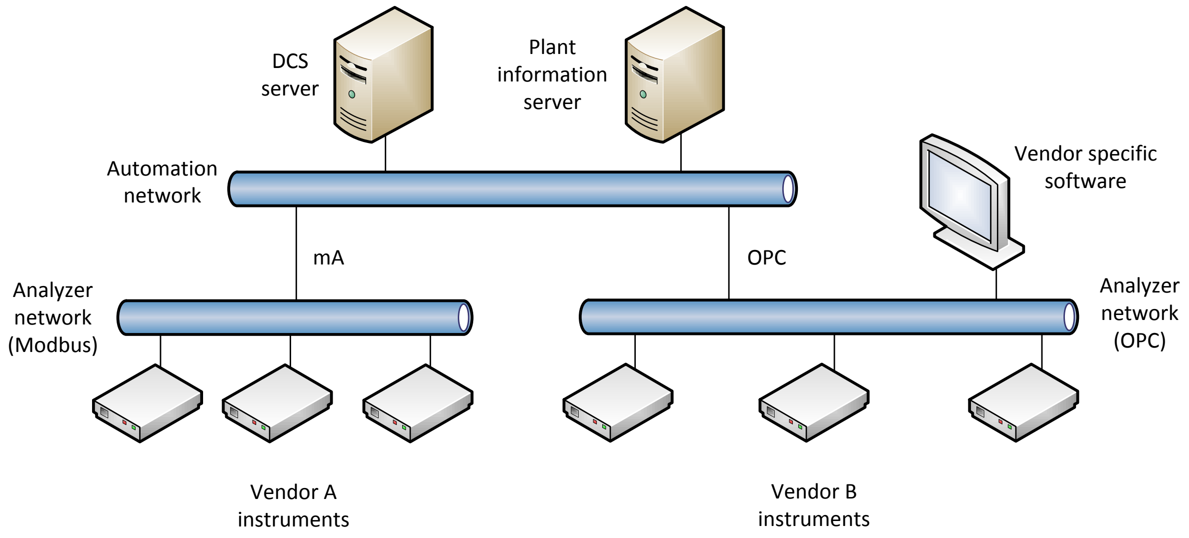


Figure 15. An analyzer network composed of different automation network types connected to a DCS network

The previous figure shows an example network where analyzer instruments transfer sample property estimates to the DCS server and plant information server where they then become available to the higher level systems. Vendor specific software might provide additional information about the instruments that rely on the proprietary information models developed by the manufacturer. Some devices may also provide additional information locally on the device that is not readily available on the plant network as these kinds of devices only provide the most basic type of information, namely the analyte properties to the higher level systems.

Analyzer devices are highly sophisticated process instruments that produce large amounts of measurement and status related data with analyte property estimates typically being the most basic type of information provided by the instruments. Additionally, instruments can provide vast amounts of diagnostic related information, for example, device status or performance indicators for the chemometric models. Therefore it would be beneficial to have a single unified view on the information provided by analyzers and also to have a standardized protocol to access this information. OPC UA and ADI specifications can be used to solve this information access problem related to analyzer devices. ADI exposes the information provided by analyzers as a unified view and OPC UA provides a standardized protocol to access this information. [51] [53] Figure 16 shows an example analyzer network based on OPC UA:

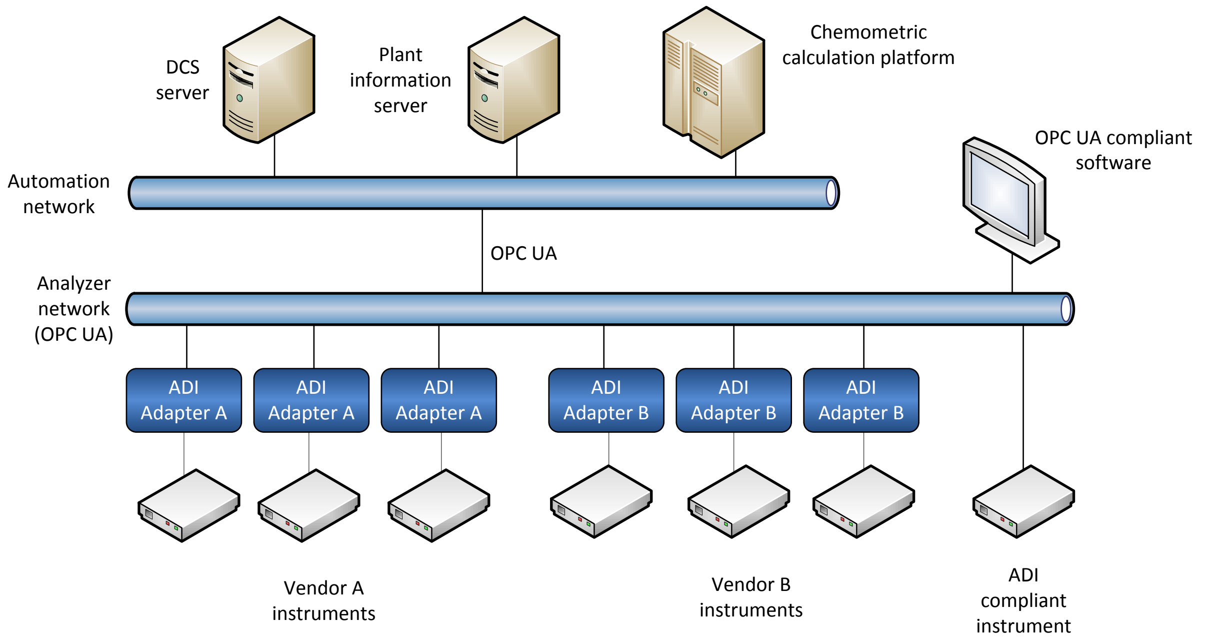


Figure 16. An analyzer network based on OPC UA providing a unified view on the analyzer information

The figure shows how all information is transferred using a single communication protocol. For higher level systems the analyzer information is provided through the OPC UA information models, whereas previously analyzer information was only available through vendor specific software or was located locally on the device. The new OPC UA implementation makes it possible for compliant applications to easily access this information and further refine it. Figure 16 shows how analyte property estimates from device chemometric models are visible to the plant information servers from ADI devices through the OPC UA protocol whilst in Figure 15 implementation was based on proprietary protocols. ADI devices not only make available the analyte properties but also other device related information and spectrum measurements accessible from the network. Figure 16 also shows an example application where a separate chemometric calculation platform is used to estimate the analyte properties from instrument spectra. This kind of implementation can take advantage of using multiple different types of data sources simultaneously and it has been previously shown that this kind of data integration can be beneficial for the chemometric models.

Many applications exist for this kind of analyzer and plant data integration, for example property estimates produced by analyzer devices can be validated using other process measurements. Further on, data from laboratory information systems (LIMS) can also be used for further data analysis and this data can be used to validate the performance of the chemometric models or even to create adaptive models [34].

# EXPERIMENTAL PART

# 5. Objectives of the experimental part

The main focus of the experimental part is the development of a data acquisition system for on-line analyzers in a very high viscosity index unit (VHVI) in the Neste Oil's Porvoo refinery. Furthermore, chemometric models are implemented to utilize these data sources. Multivariate models are used to estimate the specific properties of a target stream and these models utilize absorbance values measured with a near-infrared spectrometer. Moreover some models also utilize process data in parallel with spectral data. Emphasis is given to the implementation of the data acquisition system and the multivariate models are used as a proof-of-concept on how the data can be further processed into final estimated properties.

Ideally the data acquisition system should seamlessly connect to analyzers that rely on the OPC UA ADI specification and in addition, data acquisition from other types of data sources should also be possible. Furthermore the system is also developed in such a way that it allows the collection of historical data e.g. spectra into a database. The implemented data acquisition system fully relies on the OPC UA specification and uses the server-client model of the standard. Information models are created based on the OPC UA standard. The created analyzer specific information models are, in turn, based on the ADI companion specification.

The data acquisition system is implemented on top of Neste Jacobs' NAPCON Informer product that works as an OPC UA server with the role of this server to operate only as storage for information. All calculations related to multivariate models are performed on a separate calculation service implemented with Microsoft .NET technologies. All implementations are developed with C#-language and Microsoft .NET framework except for the historical database using a PostgreSQL database. An ADI analyzer adapter server was also required since the ABB near-infrared spectrometers used in the study did not support the OPC UA ADI specification.

# 6. Case study: Multivariate modelling of a VHVI unit data

## 6.1 Description of the process

The very high viscosity index (VHVI) unit in the Neste Oil Porvoo refinery uses heavy bottom products from other production lines and further processes these into more valuable products like lubricants and fuel oil. The production line consists of an isomerization dewaxing reactor (IDW) which is used to isomerize normal paraffins into isoparaffins. After the isomerization, the feed goes into a hydrofinishing reactor (HDF) that saturates aromatic compounds and the hydrogen is separated and recycled from the reactor outflow. The product then goes through both an atmospheric and a vacuum distillation column in order to separate the different types of products from the reactor product stream. The details of the process are not studied within the scope of this thesis, hence a highly simplified overview of the VHVI process is given below:

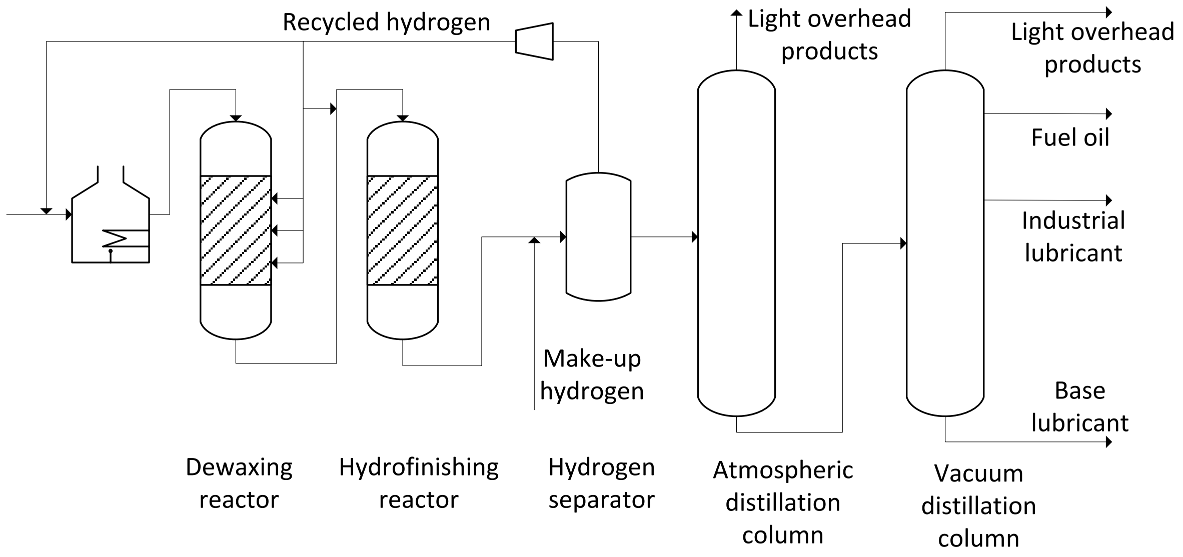


Figure 17. The very high viscosity index (VHVI) unit

An online analyzer is connected to the fuel oil, industrial lubricant and base lubricant product streams. An objective of the thesis is to create chemometric models that estimate specific properties of the product streams using the analyzer spectra and other measurements from the VHVI process. The created data acquisition system is then used to collect online data from the automation system and the analyzer network. Afterwards, this data is then further refined in a calculation platform.

## 6.2 Plant data acquisition and initial variable selection

The fuel oil product stream was used for the multivariate model case study and process measurements were selected from VHVI reactors and distillation columns. These measurements were then used to evaluate whether process measurements increase the accuracy of a chemometric model when they are used in parallel with absorbance measurements.

The process measurement data consisted of 10 minute averages that were collected over the timespan of one year between 22.7.2013 0:00 - 22.7.2014 15:10. The measurements were acquired from the refinery's information system, whilst the fuel oil analyzer spectra were gathered as separate measurement files from the plant. The fuel oil stream was selected since it contained the highest quality absorbance measurements over the selected time period. The process variables were selected by visually inspecting the piping and instrumentation diagrams of the VHVI process, whereas variables that appeared to have an influence on final product quality were selected based on process knowledge. Overall a total of 128 process measurements and calculated variables were selected and these are listed in Appendix 1. *Descriptions of the selected VHVI process measurement*. The appendix also gives detailed descriptions of the calculated variables which included finite differences, average values between variables and temperature rise between reactor beds. Calculated variables were used because the reactors contained a high number of temperature measurements from different points within the reactors. Therefore most of the calculated variables are related to the IDW and HDF reactor temperature measurements.

Five VHVI fuel oil properties were estimated using the multivariate models and corresponding laboratory test results and on-line measurements were also collected for each of these variables. The following list shows the properties that were estimated:

1. Pour point [°C]
2. Flash point [°C]
3. Cloud point [°C]
4. Density [kg/m$^3$]
5. Kinematic viscosity at 100 °C (KV100) [mm$^2$/s]

PLS1 models were created for each of the fuel oil product properties. Process measurements were collected from the plant information system and saved as Microsoft Excel files. Subsequently, these Excel and on-line analyzer files - containing the absorbance values - were read into Matlab and saved as Mat-files for easier processing. Measurement time intervals that were marked as invalid on the plant information system were removed from the files. Furthermore, the full sample was removed from the dataset if any of the sample variables were marked as an invalid.  This operation successfully removed outliers from the data which was confirmed by visual inspection of the process data. Matlab was then used to process the data and create the final multivariate models. The following figure shows an overview of the data processing methodology:

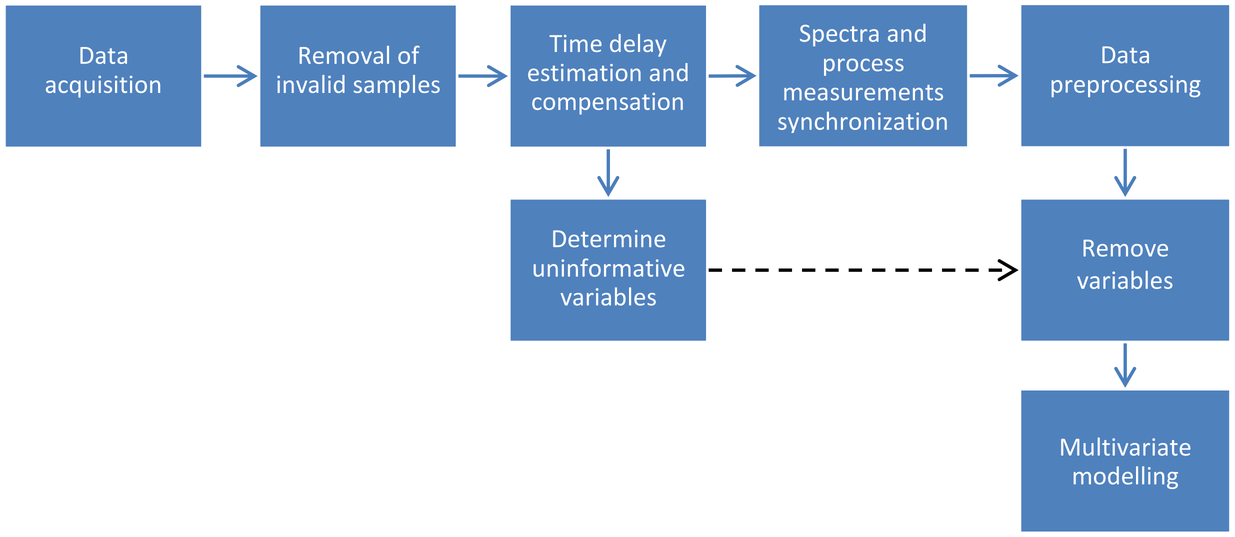

Data acquisition
Removal of invalid samples
Time delay estimation and compensation
Spectra and process measurements synchronization
Data preprocessing
Determine uninformative variables
Remove variables
Multivariate modelling

Figure 18. An overview of the data processing

## 6.3 Time delay estimation

Response delays of the estimated quantities were assessed and compensated against the process variables. Delays were corrected by using the sample based lags. Cross correlation functions (XCF) shown in Equation (41) were formed for each of the process and response variable pairs. These functions were used to estimate the integer lags for each of the variable pairs and lags were approximated using online analyzer produced estimates of the response variables. Analyzer measurements were assigned to ***y*** vector and process variable to ***x*** vector in Equation (41) throughout the lag estimation procedure.

A total of 354 response and explanatory variable pair lags were estimated. Those that had a single distinct correlation peak with below 10 sample lag were programmatically compensated, whilst other lags were estimated by manual inspection of the cross correlation function. Variable pairs that did not have a distinct correlation peak near zero lag were left uncompensated. In addition, those pairs that had the correlation peak corresponding to over 3 hours of delay were not delay compensated, which resulted in a total of 286 uncompensated variable pairs overall. The following plot shows a histogram of the estimated lags obtained for each of the response and process variable pairs:

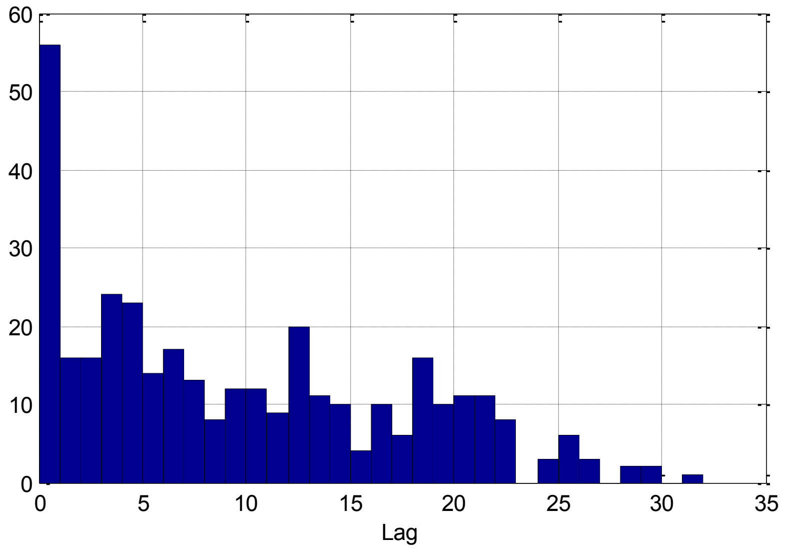


Figure 19. Histogram of the estimated lags

Figure 19 shows how some of the variables have over 20 sample lags corresponding to a time delay of over 3 hours. These are mostly related to dewaxing and hydrofinishing reactor variables which are located upstream of the VHVI unit. Variables with lower integer sample lags are related to vacuum distillation column measurements. It was noted that the responses of the estimated quantities are faster to changes in the vacuum distillation column conditions than to the conditions inside the reactors.

Laboratory result lags were also estimated since the plant information system did not contain information about when the samples were taken. Laboratory result lag estimates were determined to be accurate enough since they had very distinct cross correlation peaks. The following page lists the laboratory results sample based lags.

1. Pour point: -84
2. Flash point: -90
3. Cloud point: -78
4. Density: -74
5. KV100: -69

Laboratory result lags are negative since they are fed into the plant information system sometime after the sample is taken from the sampling point. It can be seen that the laboratory result sample based lags are surprisingly large since the samples are already 10 minute averages. This might be caused by invalid timestamps returned from the plant information system for lab measurements. These timings were deemed to be accurate enough for the data analysis carried out in this case study as true laboratory timestamps were not requested within the scope of this thesis.

## 6.4 Variable elimination

The importance of each process variable to the modeled property was evaluated after the sample lags were compensated. UVE and BVE algorithms were applied sequentially for the removal of the uninformative variables as described previously in Chapter 3.4 Selection of the significant regressors. These algorithms were repeated successively until no uninformative variables were found as shown in Figure 20 and this methodology was applied for each of the target properties by using PLS1 models. The proper number of PLS latent variables was evaluated prior to the application of these methods and the number of latent variables was selected using the $R_R$-criterion with a 0.95 cut-off limit as described in Equation (71). The selected number of latent variables was based on the point where $R_R$-metric was above or very close to the cut-off limit. The Monte Carlo method with 3 splits and 10 repeats was used for the cross validation at each of the variable elimination steps. The variable elimination methodology is shown in the following figure:

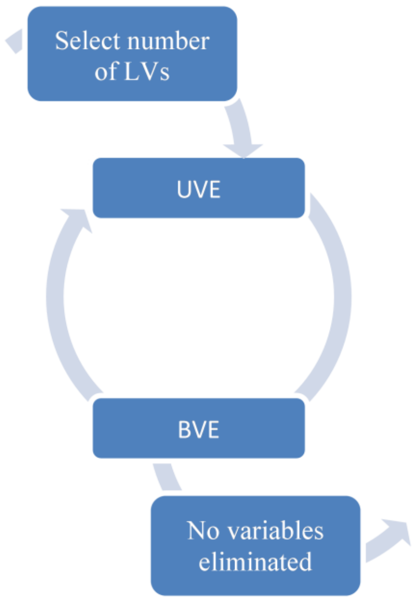


Figure 20. Variable elimination methodology

The following table shows the number of eliminated variables and used latent variables for each of the target properties:

Table 2. Number of remaining variables after the variable elimination

| | Pour point | Flash point | Cloud point | Density | KV100 |
|---|---|---|---|---|---|
| Number of LVs | 6 | 6 | 5 | 6 | 4 |
| Number of variables | 116 | 111 | 121 | 113 | 99 |

It was noted that aggressiveness of the variable elimination is very sensitive to the number of latent variables used. A higher number of process variables were eliminated when the number of latent variables was lower than that obtained in the cross-validation. This is because a model with higher number of latent variables exploits some of the useful information hidden within noisy measurements. Most of the eliminated variables were related to the calculated reactor difference variables as these variables had a large quantity of measurement noise due to the calculation of the finite differences.

Uninformative spectrum wavelengths were eliminated using the UVE algorithm with the PLS2 model. This type of model was used to evaluate importance of the wavelengths for estimating all of the product properties simultaneously. The number of latent variables was updated sequentially after every round of UVE algorithm.  The $R_R$-criterion with Monte-Carlo cross-validation was used to evaluate influential latent variables as previously and sequential updating of the model order was used because initially the criterion suggested seven latent variables. This was determined to be too high because it would result in a premature end of the elimination sequence and as a result the model order was updated after every elimination step. The number of latent variables was reduced to four during the elimination procedure, which successfully removed most of the variables belonging to the noisy and uninformative regions of the spectrum. In addition, some of the wavelength variables were removed manually prior to applying the elimination procedure which alleviated the computational burden. These manually removed variables were related to the wavelength regions containing high levels of measurement noise. Overall the number of the remaining variables was 697 from a total of 4096 wavelength variables after the manual and algorithmic elimination procedure had been applied and Figure 21 illustrates the methodology used for the wavelength variable elimination.

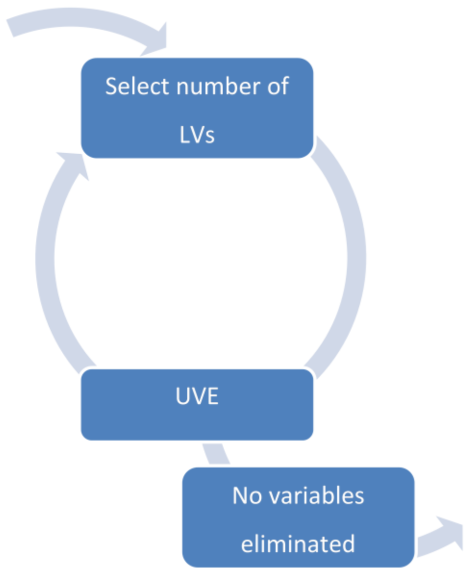


Figure 21. Wavelength variable elimination methodology

Following plot shows the wavelengths remaining after the elimination. Remaining wavelengths are related to a fundamental band and two overtone bands of the spectrum. These are the most influential wavelength regions for estimation of the fuel oil properties.

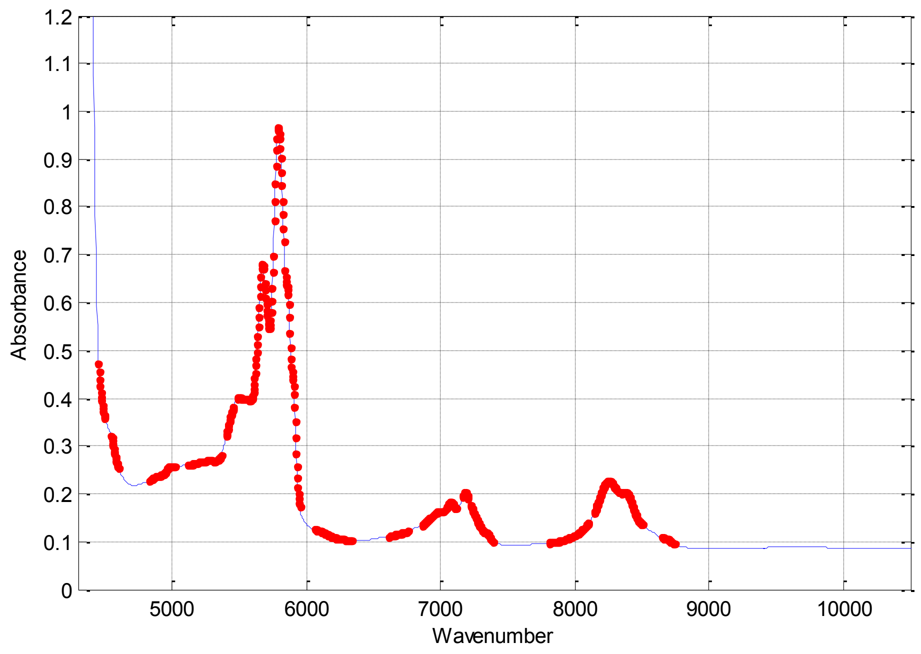


Figure 22. Remaining wavelengths after the variable elimination

Figure 22 shows the remaining wavelengths for an unprocessed spectrum. Only autoscaling was applied to the spectra before the PLS models were generated. Subsequent to this a Savitzky-Golay-Derivative filter is applied to the spectra but only during the multivariate modelling phase. Variable elimination did not involve this filter because it would amplify the noise in the spectra meaning that the elimination procedure would be too sensitive to the noise related variables and the procedure would retain some of the uninformative variables.

## 6.5 Multivariate modelling

Data was preprocessed prior to obtaining the final multivariate models. Lag compensated process measurements were first synchronized with spectrum time stamps and after this procedure a total of 303 spectra and process measurements remained for the modelling. Spectrum samples were Savitzky-Golay-Derivate filtered using a polynomial, and a derivative order of one and a window length of five data points as previously described in chapter 3.1.3 Spectral derivatives. Parameter values for the filter were determined by evaluating different parameter combinations and observing the PLS1 model performance for the different estimated properties. Figure 23 shows the original and SGD filtered spectra. As can be seen the original spectra has a clear baseline shift that was corrected by the SGD filter and which, in turn, also improved the final performance of the PLS1 models.

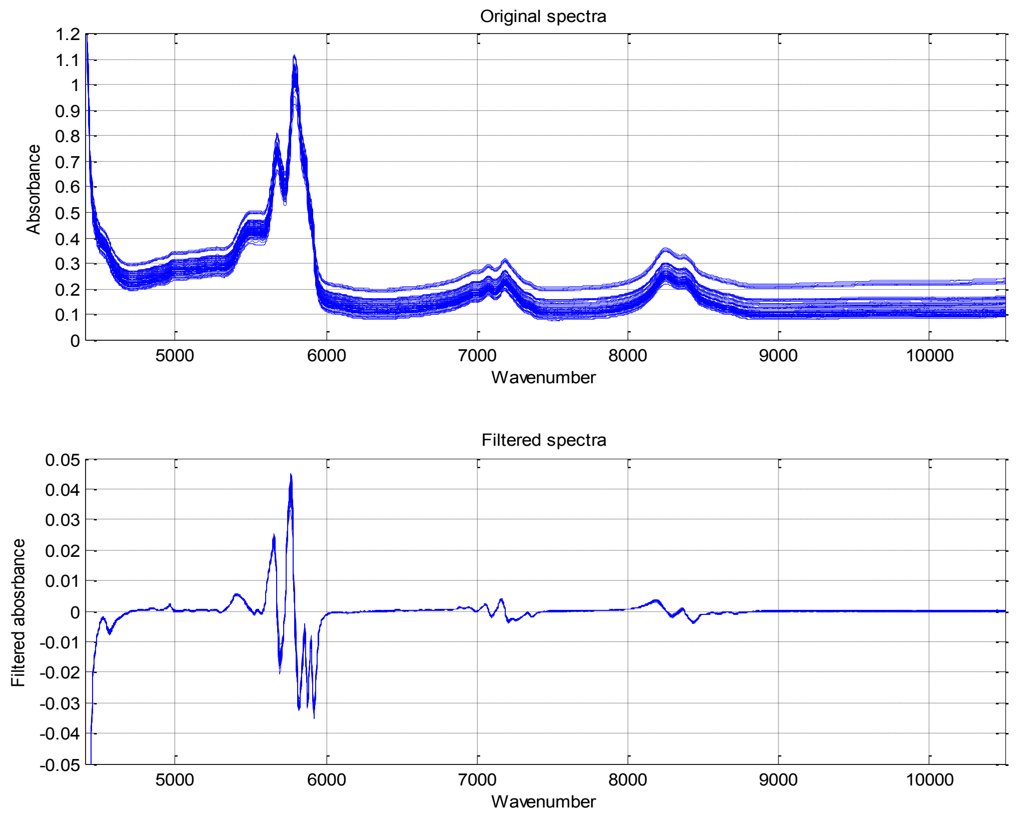


Figure 23. Original spectra and the SGD filtered spectra

Uninformative variables were removed from the full dataset after the preprocessing following the methodology outlined in chapter 6.4 Variable elimination. Autoscaling was also applied to the dataset. Finally the PLS1 models were generated using the preprocessed datasets containing only the significant variables and the NIPALS algorithm was used to obtain the PLS1 models for each of the estimated properties.

The following table shows a comparison of spectrum, process measurement and unfolded PLS1 model performances. Unfolded matrices contained the absorbance and process variables in a single matrix. Normalized *RMSEP (NRMSEP)* and *$R^2$* metrics were used to measure goodness-of-fit. The table also shows the goodness-of-fit for the analyzer produced estimates in the last column i.e. values produced by the online analyzer PLS1 models.

Table 3. Goodness-of-fit indicators for the PLS1 models

| | | Spectrum | Process | Spectrum + Process | *Online* |
|---|---|---|---|---|---|
| Pour point | LVs | 2 | 3 | 6 | *N/A* |
| | NRMSEP | 0.172 | 0.179 | 0.171 | *0.633* |
| | $R^2$ | 0.561 | 0.518 | 0.569 | *0.525* |
| Flash point | LVs | 5 | 6 | 5 | *N/A* |
| | NRMSEP | 0.163 | 0.158 | 0.155 | *0.835** |
| | $R^2$ | 0.574 | 0.605 | 0.617 | *0.553** |
| Cloud point | LVs | 6 | 3 | 5 | *N/A* |
| | NRMSEP | 0.0946 | 0.108 | 0.0899 | *0.212* |
| | $R^2$ | 0.883 | 0.843 | 0.894 | *0.869* |
| Density | LVs | 4 | 6 | 6 | *N/A* |
| | NRMSEP | 0.122 | 0.137 | 0.119 | *0.245* |
| | $R^2$ | 0.786 | 0.735 | 0.797 | *0.645* |
| KV100 | LVs | 6 | 8 | 5 | *N/A* |
| | NRMSEP | 0.0953 | 0.107 | 0.0916 | *0.146* |
| | $R^2$ | 0.877 | 0.838 | 0.884 | *0.815* |
| *Flash point online analyzer estimates were inaccurate due to large constant offset | | | | | |

Normalized *RMSEP* indicators were calculated from the ratio of *RMSEP* and range of the estimated property. *NRMSEP* and $R^2$ values were obtained by randomly drawing independent testing datasets from the full dataset. This statistical *jackknifing* technique was used due to the low number of available samples and model performance was evaluated 1000 times using 30 independent testing samples. The rest of the samples were used to generate the model and the final performance indicators in the Table 3 were calculated by taking mean value of 1000 *jackknifed* indicators. Variation of the PLS1 *NRMSEP* and $R^2$ metrics are displayed in Appendix 2. *Box plots of the PLS1 model* metrics and these box plots display the dispersion and skewness of the sampled metrics. A number of latent variables were determined by Monte Carlo cross validation and *RMSEP* curves with model order based on the minimum of *RMSEP*.

Some observations were made during the multivariate modeling. Firstly, Table 3 and PLS1 box plots in the appendix clearly show how the estimation performance is improved in all of the cases when the process data is used in parallel with the spectra. The box plots also show high statistical variation in pour and flash point $R^2$-metrics where the variation probably due to nonlinearities related to these properties. It can be also seen that the flash point model performance is better solely with the process data than in the case where spectra is used. This degraded performance with the spectra probably results from the nonlinear relations between the absorbance and the flash point. Additionally these box plots show that the cloud point and KV100 models produce the most accurate property estimates and the performance metrics related to these models also have the lowest statistical variation. Overall, each one of the cases has relatively good estimation performance when the process data is solely used for estimating the target properties. This is surprising because response of the product properties usually exhibits some level of non-linearity when using process measurement data.

All of the models benefitted from the process measurements, but the true significance of improvement is unclear due to the low number of spectral samples available. More spectral data from different operational regions of the process should be obtained in order to evaluate the true improvement in the estimation performance.

MB-PLS1 models were used to estimate each of the target properties to evaluate whether block division of the data benefits the performance of the models. These models were created using a MB-PLS NIPALS algorithm with the block score update method. Process variables were divided into 4 blocks by process equipment type and the final fifth block consisted of the spectra. The model order was determined similarly as previously in the PLS1 modeling case. Inequality presented in Equation (73) was used to test the significance of each latent variable in each of the blocks. Table 4 shows the order of the MB-PLS1 models. Goodness-of-fit metrics were evaluated by *jackknifing* technique as previously. Performance metric plots are presented in the Appendix 3. *Box plots of the MB-PLS1 model* metrics.

Table 4. Comparison of latent variables in the PLS1 and MB-PLS1 models

| | Pour point | Flash point | Cloud point | Density | KV100 |
|---|---|---|---|---|---|
| PLS1 unfold LVs | 6 | 5 | 5 | 6 | 5 |
| MB-PLS1 LVs | 6 (HDF 5) | 5 (HDF 4) | 5 (HDF 3) | 4 (HDF 3) | 6 (HDF 2) |

Table shows that the hydrofinishing block had lower number of latent variables than rest of the blocks in all of the cases. Density model also had lower number of block latent variables than the PLS1 model. Appendix 3 box plots show that the pour point estimation benefits from the block division of the data. Rest of the models do not exhibit improved estimation performance. Benefit from the block division is highly application dependent. Model performance may not benefit from the division in specific cases and therefore it is more useful in applications where it is necessary to monitor statistical metrics on each of the blocks separately.

# 7. Data processing and acquisition system

Process measurement data and analyzer produced estimates are usually located on the plant DCS servers while analyzer measurement information e.g. spectra might only be available locally on the devices. Therefore it is necessary to create a data acquisition system that integrates these multiple different data source types behind a single access point, thus making it convenient to connect to these sources and further refine this multiplicity of data into more useful information. It was shown previously in the VHVI case study that process data can improve the accuracy of the analyzer estimates and also be used to verify operational condition of the analyzer. In addition such a data acquisition system can be used to connect laboratory information systems (LIMS) in order to bring further added value to the target estimates.  This thesis focuses on creating an acquisition system that can be connected to a plant information systems and OPC UA ADI devices. Data processing software is also created to utilize these data sources and use the previously created multivariate models to further refine the data into more useful information.

## 7.1 Data acquisition system

The data acquisition system is fully based on OPC UA communication protocol and Microsoft .NET technologies. Historical data was collected into a PostgreSQL database. A system for process and spectral data acquisition was created on top of the Neste Jacobs NAPCON Informer software, which acts as an OPC UA server and was also extended to handle connections to remote OPC UA servers. These remote servers can for example be plant DCS servers and analyzer devices. OPC UA was chosen as the core technology because it allows connecting to different types of devices from different manufacturers. Complex information models can also be created and used without difficult using this specification. These kinds of information models are required to describe how the different systems interact with each other. This is especially important for cases when the multivariate models are used to refine data from a number of different information sources. Furthermore, analyzer devices are typically highly sophisticated devices that also require complex information models.

### 7.1.1 Integrating OPC UA server

Neste Jacobs OPC UA server product was extended into an integrating OPC UA server thus making it possible to integrate multiple remote information sources behind a single access point. Integrating

OPC UA server allows connection to multiple remote OPC UA servers. This system allows further refinement of the data located in multiple different types of sources into more valuable information. A client-server model was utilized to create the integrating server and the following figure illustrates the basic working principle of the integrating server:

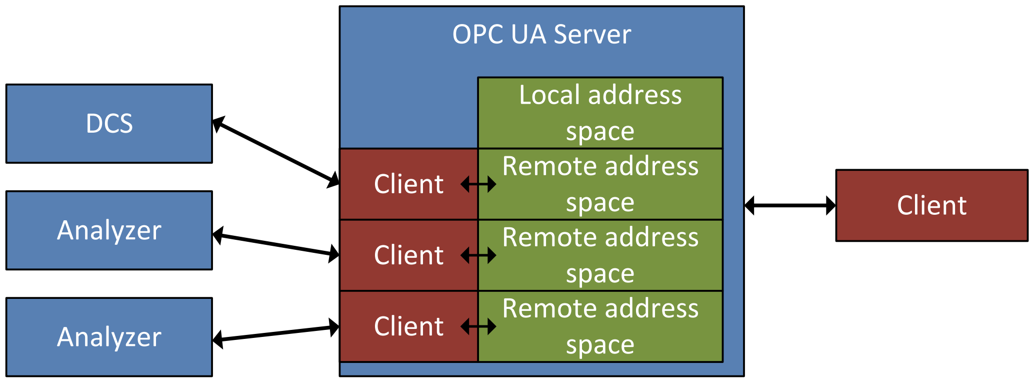


Figure 24. Integrating OPC UA server

Figure 24 uses blue to depict OPC UA servers and green to distinguish server address spaces, whilst red is used to highlight UA clients. This figure also illustrates how the DCS server and analyzer devices that act as an UA servers are mapped behind a single UA server se outside client software can then seamlessly obtain information from these data sources. It will be shown subsequently how this can be utilized to process and transfer information located within different systems. In addition Figure 24 shows how the integrating UA server contains local address space and remote address spaces. Local address space is the normal address space within a UA server, whereas remote address spaces are used to redirect the address space service requests through a UA client into the remote server. Service responses from the remote server are then forwarded to the client originally making the request. An outside client does not need to care about what information is located on which systems as all information in the remote servers can be seamlessly accessed from the integrating server. To make these possible, certain mappings must be performed inside the integrating server in order to avoid possible conflict situations. Such conflicts might arise if the remote servers or the integrating server use the same namespace names between the different address spaces. Conflict situation is avoided by namespace mappings inside the integrating UA server i.e. remote namespaces are mapped by prepending address of the remote server to the namespace. This kind of namespace mapping is also performed in Unified Automation UaGateway that connects to remote UA servers

[56]. The UaGateway is a wrapper application that allows OPC UA clients to connect to legacy OPC servers. Additionally, this application can also be used as a proxy to other OPC UA servers. The difference between the UaGateway and the integrating server that was created is that the integrating server not only proxies the remote UA servers but also acts as a data repository and has its own local address spaces. An outline of the integrating servers' namespace mapping is illustrated below:

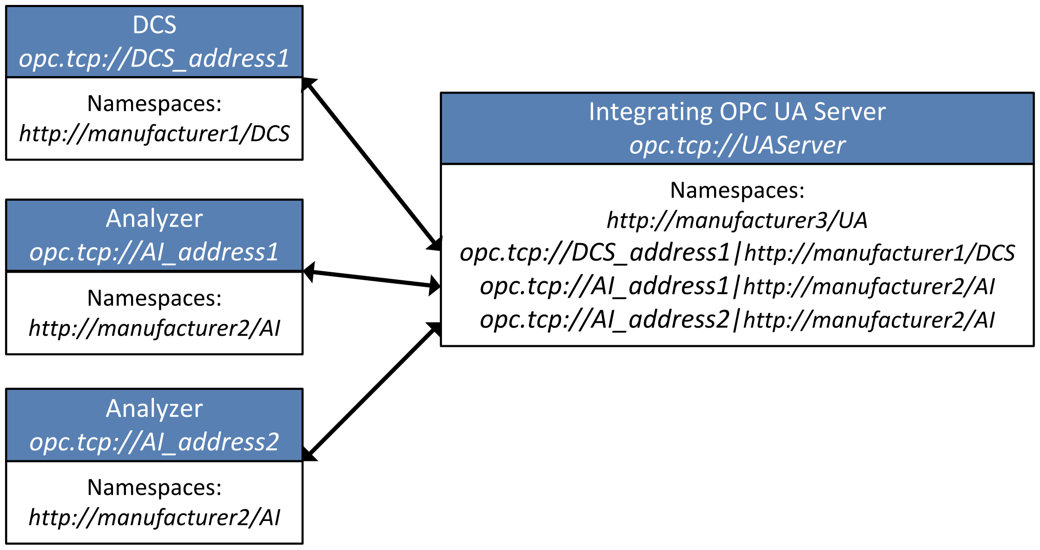

DCS
*opc.tcp://DCS_address1*
Namespaces:
*http://manufacturer1/DCS*
Analyzer
*opc.tcp://AI_address1*
Namespaces:
*http://manufacturer2/AI*
Analyzer
*opc.tcp://AI_address2*
Namespaces:
*http://manufacturer2/AI*
Integrating OPC UA Server
*opc.tcp://UAServer*
Namespaces:
*http://manufacturer3/UA*
*opc.tcp://DCS_address1|http://manufacturer1/DCS*
*opc.tcp://AI_address1|http://manufacturer2/AI*
*opc.tcp://AI_address2|http://manufacturer2/AI*

Figure 25. Namespace mapping inside the integrating UA server

The example configuration above shows how two analyzer devices from the same manufacturer contain the same namespace, a conflict that would cause integrating server to have no way to determine where *NodeId* based UA service requests should be redirected. This problem is avoided by proper namespace mapping inside the integrating server i.e. by performing namespace mapping when the integrating server creates the connection to the remote server. The mapped namespace is then added to the namespace list inside the integrating server. Namespace mappings are updated automatically if namespaces inside any of the remote servers change. The OPC UA specification states that namespaces should not be removed from the server namespaces during the lifetime of the server, therefore remote namespaces are not removed from the integrating server if connection is lost to the remote server. Service requests to the unreachable servers are handled and proper error codes are returned to the client making the request. Namespace mapping is also performed on read and write requests to the UA *Variables* that contain *NodeId* or *ExpandedNodeId* as their values. In addition *Array* and *Complex* type values are also checked for possible node ids.

It was also necessary to implement services for UA *Reference* creation between remote-local and remote-remote address spaces. These references are later used, for example, to specify multivariate model data inputs and outputs between the servers' address spaces. Reference creation or modify requests are not redirected to the remote servers because such references made over different systems would be invalid from the remote server point of view. References are instead created inside the local address space of the integrating server.

The server also handles request redirect loops that can be caused by a circular configuration of integrating servers. For example a UA *Query* request would cause a redirect loop that would never finish in the circular configuration. Redirect loops are therefore handled inside the integrating server by adding a globally unique identifier (GUID) to a *RequestHeader AuditEntryId* [51] for each of the redirected requests. A list of these GUIDs is tracked inside the integrating server and the redirect loop is detected when an incoming service request has a GUID that is already in the list of the GUIDs. The redirect loop is then gracefully terminated by immediately returning an empty result or error code from the service.

An abstract representation of the remote address space functionality is presented in Appendix 4. *Representation of the remote address space* services. The figure in the appendix also depicts the history functionality of the remote address spaces and this service is described in more detail within chapter 7.1.4 History database. The same figure also shows that remote read and write services have the simplest implementations as these services only require namespace mapping to the requests and results. Other services have a more complex implementation, for example, the browse service has to resolve possible locally defined UA references. These references can target the nodes inside the remote address space, or the local address space nodes from the remote ones.

### 7.1.2 Data acquisition from analyzer devices

The literature part of the thesis introduced the OPC UA ADI specification that is a standardized information model used for analyzer devices. This specification defines how the measurement and diagnostic related information is presented inside the analyzer devices. Moreover it also specifies how certain maintenance related procedures are invoked on the device. However, this thesis only focuses on the data acquisition parts of the ADI specification for spectrometric analyzer devices and Figure 14 outlines the relevant ADI specification terms used in this chapter.

ADI object model is created by subtyping *AnalyserDeviceType*, *AnalyserChannelType*, *StreamType*, *AccessorySlotType* and *AccessoryType* type nodes. The *AnalyserDeviceType* is subtyped for each of the different analyzer types. ADI specification specifies *AnalyserDeviceType* subtypes for different instrument types. Instrument vendors can also extend these types to expose some custom parameters. Figure 26 shows subtypes of the *AnalyserDeviceType* as defined in ADI specification.

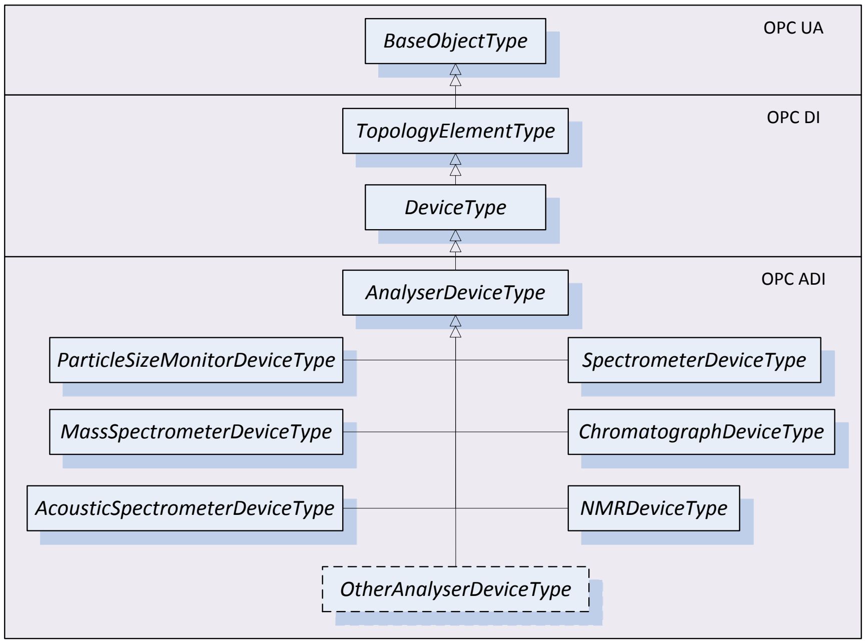


Figure 26. Subtypes and super types of the *AnalyserDeviceType*

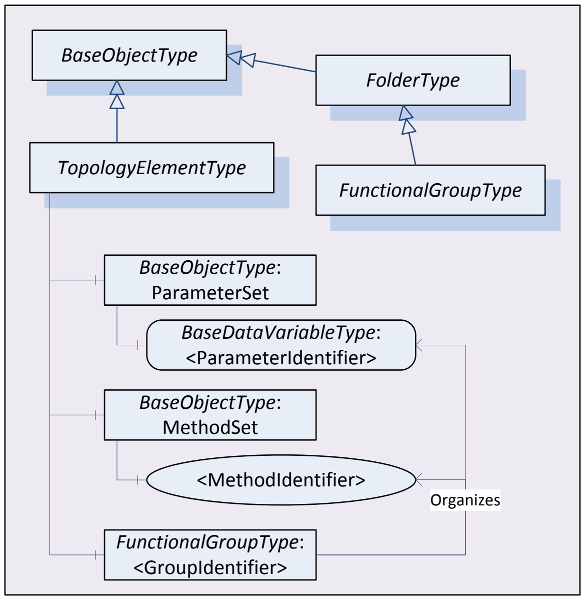

BaseObjectType
FolderType
TopologyElementType
FunctionalGroupType
BaseObjectType: ParameterSet
BaseDataVariableType: <ParameterIdentifier>
BaseObjectType: MethodSet
<MethodIdentifier>
Organizes
FunctionalGroupType: <GroupIdentifier>

Figure 27. OPC UA notation diagram for the *TopologyElementType*

Figure 26 shows that the *AnalyserDeviceType* is a subtype of a *DeviceType* and a *TopologyElementType*. These two types are specified in an OPC UA companion specification for devices (OPC DI). The ADI *AnalyserDeviceType* and the DI *DeviceType* type definitions are specified so that they are as flexible as possible but they also give a concrete and unified view of the information contained inside the devices. The *TopologyElementType* is used to present device properties and methods in an organized structure and a definition of this type is presented in Figure 27. [57]

Figure 27 shows *ParameterIdentifiers*, *MethodIdentifiers* and *GroupIdentifiers* which are the device specific properties, methods and functional groups. The *ParameterSet* and the *MethodSet* contain all the parameters and the methods belonging to the subtype of the *TopologyElementType*. In addition the *FunctionalGroupTypes* are used to further organize the device parameters and methods into logical groups.

The *DeviceType* contains read-only parameters related to the device model and the manufacturer information including the software version and serial number of the device. Device documentation and the status of the device can also be presented under the *DeviceType* instances. The DI specification states that some of these parameters are mandatory and must be implemented by the device vendor, however, the vendor has the option not support these mandatory parameters by setting the variables to default values. Some of the device information is also considered as read-only, for example, some of the hardware and software related parameters are read-only. These parameters can only change after a software or configuration update of the device. The *DeviceType* in DI specification is an abstract type and therefore no direct instances of the type can exist inside the server address space. In contrast, the ADI specification extends the definition of the DI *DeviceType* into a more concrete *AnalyserDeviceType*.

The *AnalyserDeviceType* inherits the *DeviceType* parameters and extends it with analyzer specific information. Analyzer device information model is partly presented in Figure 28. *StreamType* instances under the *AnalyserChannels* are used to read the measurement related information from the analyzer device and as a result the *StreamType* is considered to be the most important type definition for this thesis. The *AnalyserDeviceType* also has parameters and methods that can be used for obtaining information about the status and configuration of the analyzer. Moreover analyzer channels can be also controlled through the methods provided by *AnalyserDeviceType*. Additional methods for the configuration and maintenance of the device also exist but are not presented within the scope of this thesis.

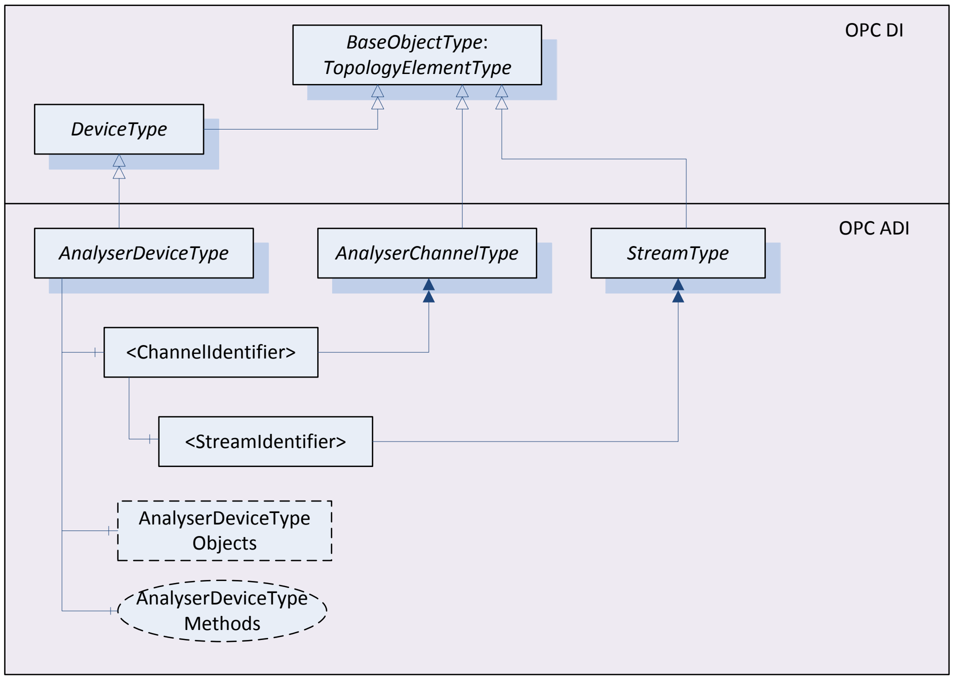


Figure 28. Hierarchy of the *AnalyserDeviceType*, *AnalyserChannelTypes* and *StreamTypes*

Figure 28 shows that *the AnalyserDeviceType* may have multiple *HasComponent* references to *AnalyserChannelType* instances representing the instrument channels meaning that multiple analyzer channels can be active at once. The *AnalyserChannelType* contains all the necessary parameters for reading the active state of the channel and it also contains methods for changing the state of the channel, for example, to enable or disable the channel data acquisition. Furthermore, the *AnalyserChannelType* contains configuration specific parameters and maintenance specific methods like those in the *AnalyserDeviceType*. The *AnalyserChannelType* instances may have multiple child *StreamType* instances and certain parameters can indicate which of the child streams is currently active. The *StreamType* instances are used for analyzer data acquisition and each of the different analyzer types has different characteristics for the streams. As a result the *StreamType* in ADI specification is denoted as an abstract type and therefore the *StreamType* must be subtyped for each different device type. The ADI specification specifies some common analyzer stream types and Figure 29 displays the subtypes of the *StreamType* as given in the ADI specification.

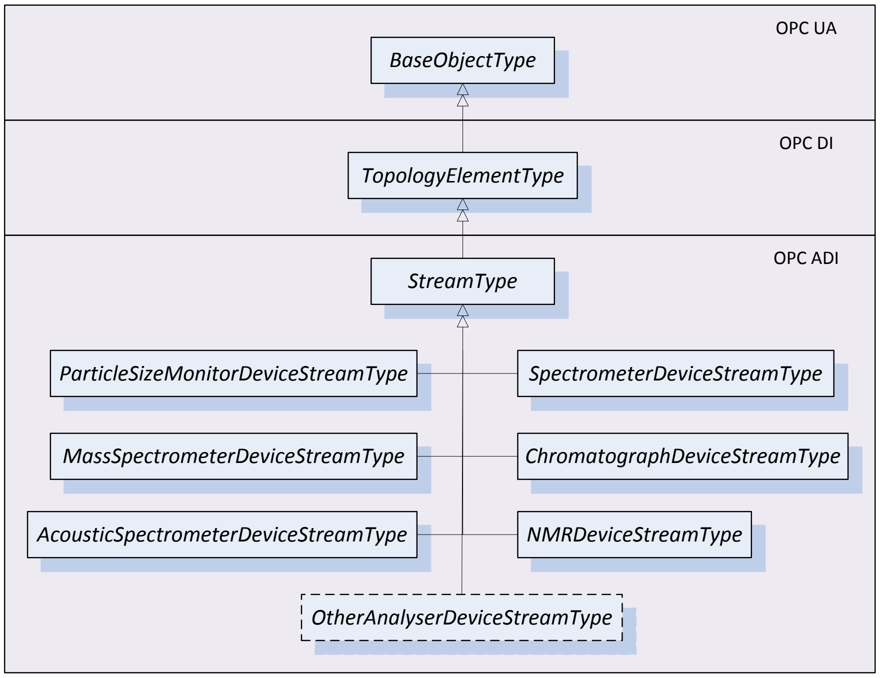

OPC UA
BaseObjectType
OPC DI
TopologyElementType
OPC ADI
StreamType
ParticleSizeMonitorDeviceStreamType
SpectrometerDeviceStreamType
MassSpectrometerDeviceStreamType
ChromatographDeviceStreamType
AcousticSpectrometerDeviceStreamType
NMRDeviceStreamType
OtherAnalyserDeviceStreamType

Figure 29. Subtypes of the *StreamType*

This thesis focuses only on spectrometric instruments. Therefore the *SpectrometerDeviceStreamType* is outlined in more detail which is the most relevant type from data acquisition point of view. Definition of the base *SteamType* from Figure 29 is given in the table below:

Table 5. Definition of the *StreamType*

| **References** | **NodeClass** | **BrowseName** | **TypeDefinition** |
|---|---|---|---|
| Inherits structure of *TopologyElementType* as defined in Figure 27 | | | |
| HasComponent | Object | Configuration | FunctionalGroupType |
| HasComponent | Object | Status | FunctionalGroupType |
| HasComponent | Object | AcquisitionSettings | FunctionalGroupType |
| HasComponent | Object | AcquisitionStatus | FunctionalGroupType |
| HasComponent | Object | AcquisitionData | FunctionalGroupType |
| HasComponent | Object | ChemometricModelSettings | FunctionalGroupType |
| HasComponent | Object | Context | FunctionalGroupType |

The *StreamType* is extended into a *SpectrometerDeviceStreamType* by extending the *FunctionalGroupTypes* under the *StreamType*. The following table shows the extended *Configuration* object denoted in Table 5 of the *SpectrometerDeviceStreamType*:

Table 6. *SpectrometerDeviceStreamType Configuration* parameters

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| Organizes | Variable | IsEnabled | Boolean | DataItemType | Yes |
| Organizes | Variable | IsForced | Boolean | DataItemType | No |
| Organizes | Variable | ActiveBackground | Float | YArrayItemType | Yes |
| Organizes | Variable | ActiveBackground1 | Float | YArrayItemType | No |

*IsEnabled* Boolean variable indicates whether the stream can be used for the data acquisition, whilst the *IsForced* parameter indicates whether the current stream is the only stream that can be used for the data acquisition in the current channel. The *ActiveBackground* is the background spectrum used for spectrometric measurements and *ActiveBackground1* parameter is used in spectrometers that require black and white backgrounds. In these devices the *ActiveBackground* is the white background and *ActiveBackground1* is the black background. Some devices may also require more background spectra, these are denoted as a *ActiveBackground<n>* where *n* is a running number. The background spectra are always obtained during the calibration phase of the device. Data types of the spectra variables and parameters are a *YArrayItemTypes* which contain the measurement values and x-axis points. In addition the measurement unit and range are also indicated inside the data type.

Following table shows the extended parameters of the *Status FunctionalGroupType* which was denoted in Table 5.

Table 7. *SpectrometerDeviceStreamType Status* parameters

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| Organizes | Variable | DiagnosticStatus | DeviceHealthEnumeration | DataItemType | Yes |
| Organizes | Variable | LastSampleTime | DateTime | DataItemType | Yes |
| Organizes | Variable | LastCalibrationTime | DateTime | DataItemType | No |
| Organizes | Variable | LastValidationTime | DateTime | DataItemType | No |

The *Status* object has two mandatory variables associated with it. The first one indicates the diagnostic status of the stream, whilst the second one indicates the time stamp of the latest sample. The calibration and validation times of the stream are optional parameters.

The following table shows the extended parameters of the *AcquisitionSettings FunctionalGroupType* which was denoted in Table 5*:*

Table 8. *SpectrometerDeviceStreamType AcquisitionSettings* parameters

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| Organizes | Variable | TimeBetweenSamples | Duration | AnalogItemType | No |
| Organizes | Variable | SpectralRange | Range | DataItemType | No |
| Organizes | Variable | Resolution | Enum/Float | DataItemType | No |
| Organizes | Variable | RequestedNumberOfScans | Int32 | AnalogItemType | No |
| Organizes | Variable | Gain | Enum/Float | DataItemType | No |
| Organizes | Variable | TransmittanceCutoff | Range | DataItemType | No |
| Organizes | Variable | AbsorbanceCutoff | Range | DataItemType | No |

All of the parameters under the *AcquisitionSettings* are optional. Therefore it is up to the vendor to decide whether to support the listed parameters. The *TimeBetweenSamples* indicates number of milliseconds between two consecutive starts of acquisition, whilst the *SpectralRange* defines the spectral range of the acquisition which also indicates the unit of the spectral measurement as an *EngineeringUnits* property. The *RequestedNumberOfScans* indicates the number of averaged scans used for the current spectral measurement. *Gain*, *TransmittanceCutoff* and *AbsorbanceCutoff* parameters are the detector gain, transmittance and absorbance clipping limits for the current measurement respectively.

Table 9 shows the extended parameters of the *AcquisitionStatus FunctionalGroupType:*

Table 9. *SpectrometerDeviceStreamType AcquisitionStatus* parameters

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| Organizes | Variable | IsActive | Boolean | DataItemType | No |
| Organizes | Variable | Progress | Float | DataItemType | No |
| Organizes | Variable | ExecutionCycle | ExecutionCycleEnumeration | DataItemType | No |
| Organizes | Variable | ExecutionCycleSubcode | UInteger | MultiStateDiscreteType | No |
| Organizes | Variable | NumberOfScansDone | Int32 | AnalogItemType | No |

*IsActive* parameter indicates whether acquisition is currently active in the stream, whilst the progress value indicates the progress of the acquisition as a percentage from the completion. *ExecutionCycle* and *ExecutionCycleSubcode* parameters are used to indicate the current state of the acquisition cycle, whilst the *NumberOfScansDone* indicates number of scans that are ready in the currently running acquisition.

*StreamTypes AcquisitionData* contains the actual measurement data obtained from the spectrometric measurement. The definition for this type is shown in the table below:

Table 10. *SpectrometerDeviceStreamType AcquisitionData* parameters

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| Organizes | Variable | RawData | Float | YArrayItemType | No |
| Organizes | Variable | ScaledData | Float | YArrayItemType | Yes |
| Organizes | Variable | AcquisitionCounter | Counter | AnalogItemType | Yes |
| Organizes | Variable | AcquisitionResultStatus | AcquisitionResultStatusEnumeration | DataItemType | Yes |
| Organizes | Variable | Offset | Duration | AnalogItemType | No |
| Organizes | Variable | AcquisitionEndTime | DateTime | DataItemType | Yes |
| Organizes | Variable | TotalNumberOfScansDone | Int32 | AnalogItemType | Yes |

| | | | | | |
|---|---|---|---|---|---|
| Organizes | Variable | BackgroundAcquisitionTime | DateTime | DataItemType | Yes |
| Organizes | Variable | PendingBackground | Float | YArrayItemType | Yes |
| Organizes | Variable | PendingBackground1 | Float | YArrayItemType | No |
| Organizes | Variable | <ProcessVariableIdentifier> | | ProcessVariableType | No |

*RawData* represents the raw measurement data obtained by the instrument which has arbitrary units, whilst *ScaledData* contains the actual absorbance or transmittance obtained during the acquisition phase. The *AcquisitionCounter* is an incremented value that increases after every sample and the *AcquisitionResultStatus* shows the status of the last measurement which is used to indicate whether the measurement was successful. The *Offset* parameter reveals the difference in milliseconds from start of the sample extraction to the start of the analysis, whereas *AcquisitionEndTime* represents end time of the acquisition. The *TotalNumberOfScansDone* indicates how many scans were performed for the current acquisition and *BackgroundAcquisitionTime* represents the time when the background spectrum used for this acquisition was acquired. The *PendingBackground* parameter represents the latest white background spectrum acquired by this stream although this is not necessarily used as a background spectrum. Instead, the *ActiveBackground* inside the *Configuration FunctionalGroupType* represents the background spectrum currently in use. In addition, the *PendingBackground1* designation is used when an instrument requires white and black background spectra. Additional *PendingBackground* parameters are represented as the *PendingBackground<n>* variables. It must be also noted that all variables under the *AcquisitionData* are read-only parameters. The *AcquisitionData FunctionalGroup* may also contain references to the process variables that represent the estimated response variables whose type definitions are *ProcessVariableTypes*. This is used to provide a stable address space view to the response variable and it has a single *HasDataSource* reference to the estimated variable. Values for these process variables are updated after chemometric model is applied to the *ScaledData*.

The *StreamTypes ChemometricModelSettings* parameter has *HasComponent* references to the chemometric models used for the stream. The model instances referenced by *ChemometricModelSettings* and are defined as *ChemometricModelType*. Overall these chemometric model types are optional. Table 11 shows the definition of the *ChemometricModelType*.

Table 12. Definition of the *ChemometricModelType*

| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Mandatory** |
|---|---|---|---|---|---|
| HasProperty | Variable | Name | LocalizedText | PropertyType | Yes |
| HasProperty | Variable | CreationDate | DateTime | PropertyType | Yes |
| HasProperty | Variable | ModelDescription | LocalizedText | PropertyType | Yes |
| HasInput | Variable | <UserDefinedInput> |  | BaseVariableType | Yes |
| HasOutput | Variable | <UserDefinedOutput> |  | BaseVariableType | Yes |

The *Name* parameter indicates name of the chemometric model and this may represent, for example, the measured quantity with the model version number. The *CreationDate* parameter is the creation date of the model and the *ModelDescription* contains the description of the model. *ChemometricModelType* instances also contain references to the inputs and to the outputs of the model. The inputs are the explanatory variables used in the model which are denoted using *HasInput* references and these references may point, for example, to the *ScaledData* parameter of the analyzer stream. In contrast, the output references point to nodes which are the response variables of the chemometric model. Such *HasInput* and *HasOutput* reference types are subtypes of *HasOrderedComponent* reference type and this requires that the server be consistent on the order of the nodes returned when a source node is browsed.

The ADI specification also states that the *ChemometricModelType* is a *Variable NodeClass*. Its *DataType* is *ByteString* and the value contains a binary blob representation of the chemometric model. Instances of the *ChemometricModelType* may not be created since the type is marked as an abstract and, as such, vendor has to subtype the *ChemometricModelType*. [53]

### 7.1.3 Data acquisition from the VHVI ABB spectrometers

An ABB on-line spectrometer was used to measure VHVI fuel oil product properties and an ADI wrapper server was created for the analyzer in order to enable data acquisition from the device since the instrument did not support OPC UA ADI specification. The analyzer produces binary measurement files to the hard disk of the analyzer computer and these files are organized into folders containing the measurements for each of the analyzer streams. The Analyzer binary files are converted into a readable text files using proprietary converter software before the wrapper server parses these text files and presents the necessary information inside the server address space through the OPC UA information models. The wrapper server is configured using a XML-file that defines the stream measurement file folder and what information is presented inside the address space from the converted files. A file system monitor is used to detect when new files are written to the configured folders. These files are then automatically converted and presented to the server address space. The figure below illustrates the working principle of the ADI wrapper server:

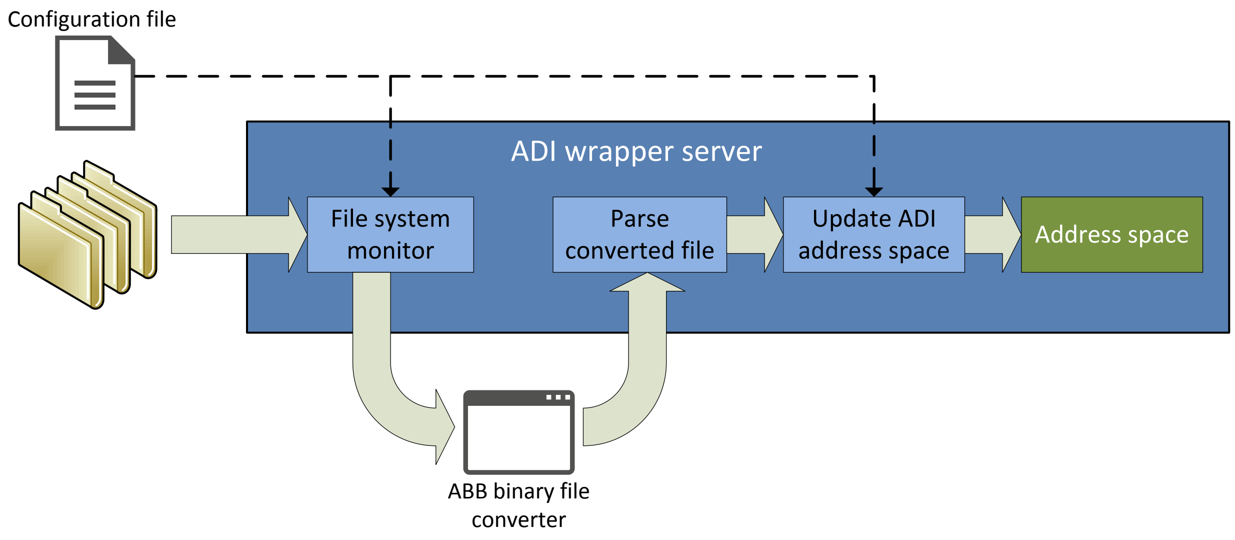


Figure 30. Working principle of the ADI wrapper server

The converted files contain a limited amount of *well-defined* information that can be directly presented using the ADI information model. The following lists the information that is presented using the ADI *SpectrometerDeviceStreamType* information model:

- *Configuration - IsEnabled*
- *AcquisitionData - ScaledData*
- *AcquisitionData - AcquisitionResultStatus*
- *Status - LastSampleTime*
- *Status - DiagnosticStatus*
- *AcquisitionSettings - Gain*
- *AcquisitionSettings - Resolution*
- *Context - UserId*

Status related information is determined using the detector saturation and analog/digital-converter overflow information on the converted file. Non-ADI-applicable information is then presented using an *ABBAdapterSpectrometerDeviceStreamType* that extends the *SpectrometerDeviceStreamType*. The information model for this type is presented next.

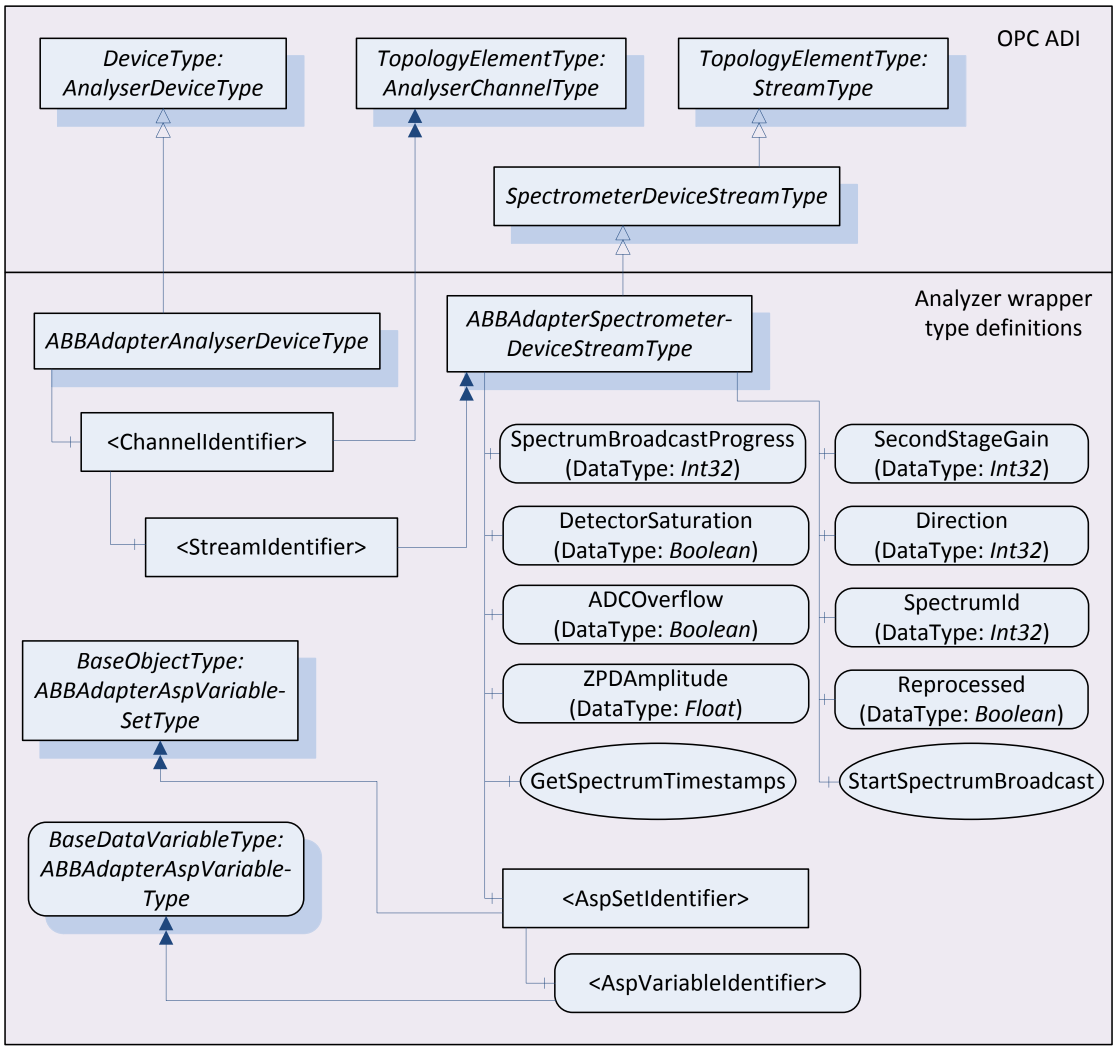

OPC ADI
DeviceType: AnalyserDeviceType
TopologyElementType: AnalyserChannelType
TopologyElementType: StreamType
SpectrometerDeviceStreamType
Analyzer wrapper type definitions
ABBAdapterAnalyserDeviceType
ABBAdapterSpectrometer-DeviceStreamType
<ChannelIdentifier>
<StreamIdentifier>
SpectrumBroadcastProgress (DataType: Int32)
DetectorSaturation (DataType: Boolean)
ADCOverflow (DataType: Boolean)
ZPDAmplitude (DataType: Float)
SecondStageGain (DataType: Int32)
Direction (DataType: Int32)
SpectrumId (DataType: Int32)
Reprocessed (DataType: Boolean)
GetSpectrumTimestamps
StartSpectrumBroadcast
BaseObjectType: ABBAdapterAspVariable-SetType
BaseDataVariableType: ABBAdapterAspVariable-Type
<AspSetIdentifier>
<AspVariableIdentifier>

Figure 31. ABB analyzer wrapper information model

The *ABBAdapterSpectrometerDeviceStreamType* contains *<AspSetIdentifier>* objects and *<AspVariableIdentifier>* variables that are dynamically created to reflect measurement file contents and wrapper server configuration. These dynamically added nodes are needed since the information inside the ABB measurement files has no *well-defined* structure. A wrapper server configuration defines what information is dynamically read to the address space. This information is also ABB analyzer device configuration dependent and it contains, for example, preprocessing related data.

The wrapper server contains a specialized service for *broadcasting* spectrum measurement files from the hard disk of the analyzer. This service works by presenting the spectrum measurement information within the server address space on file-by-file basis and in this way any missing data can be streamed onto the history database through an integrating OPC UA server as is shown later on. Broadcasting is initiated by a client first requesting the timestamps of the spectrum measurements by calling the *GetSpectrumTimestamps* method. The returned timestamps are then compared against the history database contents any spectrum files whose timestamps are missing from the history database can then be streamed from the device into the database by calling *StartSpectrumBroadcast*. In addition a *SpectrumBroadcastProgress* variable is used to indicate the progress of the broadcast service. Furthermore, the wrapper server also extends the type definitions for chemometric models and response variables which are presented in the following figure.

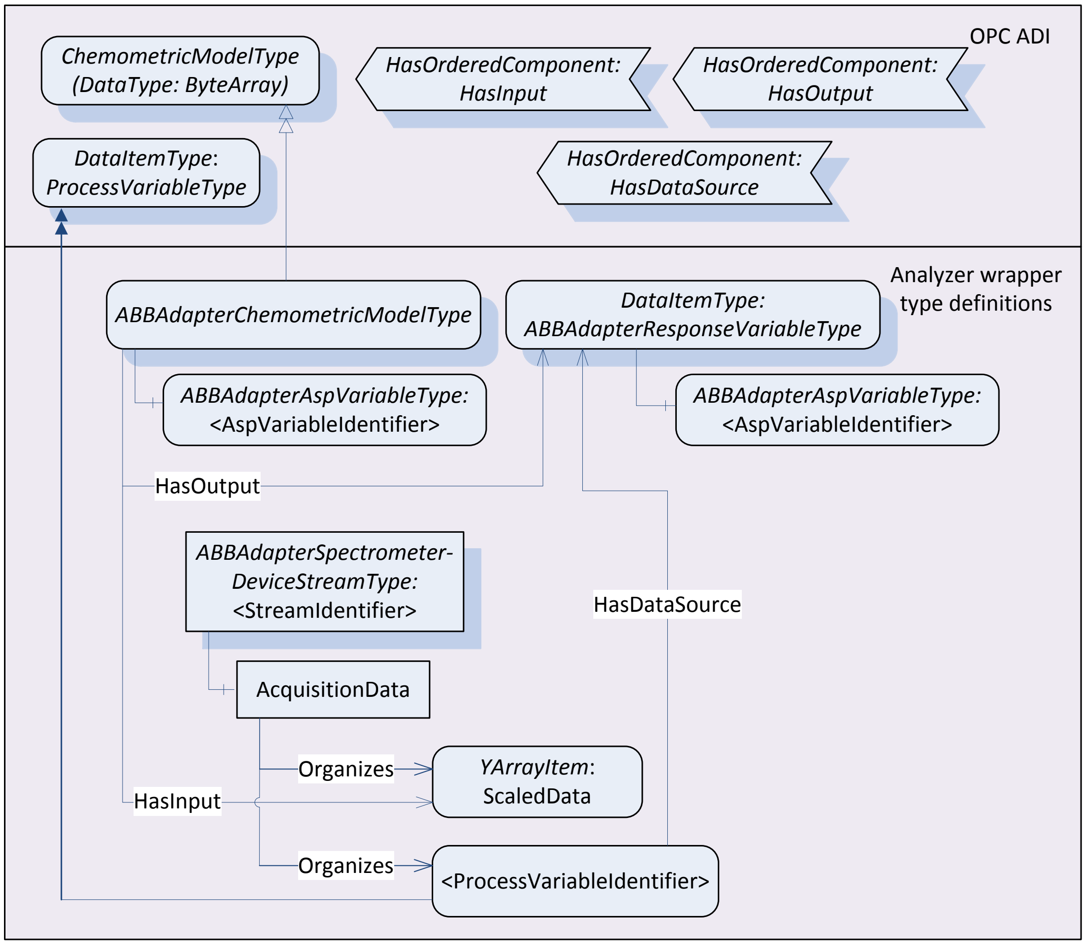

OPC ADI
ChemometricModelType (DataType: ByteArray)
HasOrderedComponent: HasInput
HasOrderedComponent: HasOutput
DataItemType: ProcessVariableType
HasOrderedComponent: HasDataSource
Analyzer wrapper type definitions
ABBAdapterChemometricModelType
DataItemType: ABBAdapterResponseVariableType
ABBAdapterAspVariableType: <AspVariableIdentifier>
ABBAdapterAspVariableType: <AspVariableIdentifier>
HasOutput
ABBAdapterSpectrometer-DeviceStreamType: <StreamIdentifier>
HasDataSource
AcquisitionData
Organizes
YArrayItem: ScaledData
HasInput
Organizes
<ProcessVariableIdentifier>

Figure 32. Chemometric model and response variable type definitions for the ADI wrapper

Figure 32 also shows how data flow relations are modelled inside the wrapper server based on the ADI specification. *HasOutput* references are used to indicate the *ABBAdapterChemometricModelType* response variables. This chemometric model type contains dynamically created *AspVariableIdentifiers* and these nodes are dynamically generated as in an analyzer adapter stream type. A target of the *HasOutput* reference is the *ABBAdapterResponseVariableType* that also contains dynamically added *AspVariableIdentifier* components. A *ScaledData* component of the *SpectrometerDeviceStreamType* is used as input for the wrapper chemometric model and this relation is represented by a *HasInput* reference type.

### 7.1.4 History database

A history database is used to save archived measurements and other types of historical data. The NAPCON Informer product uses a PostgreSQL relational SQL database to collect the archived data. The existing database models were extended to accept analyzer specific data including the vector types for spectrum samples and complex OPC UA binary encoded data. History data collection from remote OPC UA servers was implemented on the integrating OPC UA server and the following figure gives a schematic overview of the history data collection functionality of the integrating server:

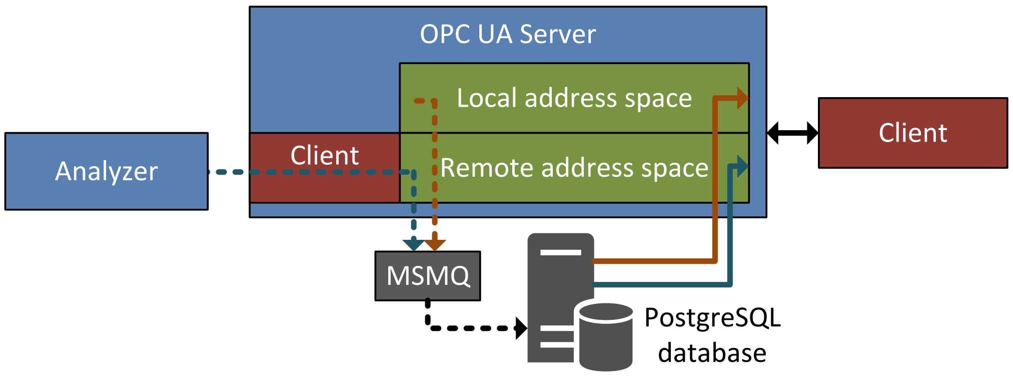


Figure 33. Integrating OPC UA server and the history database

Figure 33 shows how individual value changes, depicted as dashed arrows, are registered to the SQL database through a Microsoft message queue (MSMQ). The history read services - depicted as solid arrows - are seamlessly available on the different address spaces and MSMQ is used for queuing SQL write instructions. Archiving of the remote data sources is implemented using OPC UA client *MonitoredItems*. Variable changes are detected within the remote address spaces and new values are written to the history database. It was necessary to implement this feature since OPC UA analyzer devices do not usually support history services. As a result it was possible, for example, to archive absorbance vectors to the SQL database from the ADI information models. Additionally, remote address spaces also support direct relaying of history read requests to the remote servers, which can be used, for example to, read history data from LIMS systems in future applications.

Database SQL models were also extended to support ADI specific data and the following figure shows an overview of the database models used for ADI related archiving.

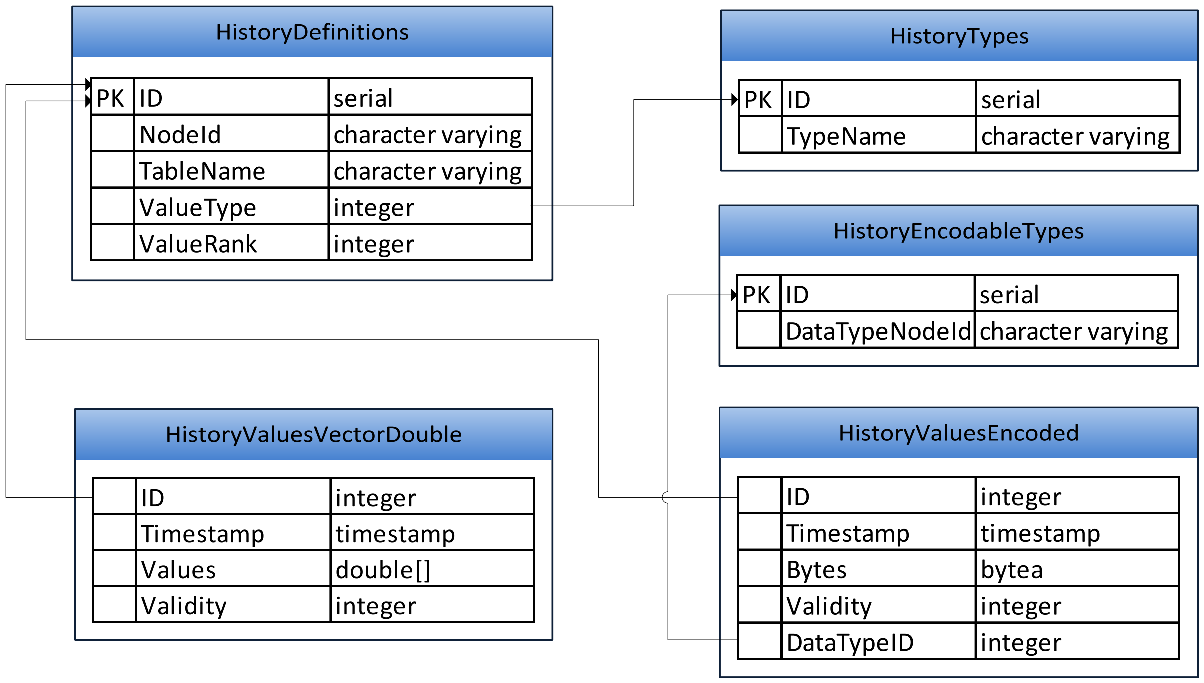


Figure 34. An SQL model for the ADI specific parts of the history database

*HistoryDefinitions* stores information about archived OPC UA nodes. *NodeId* table column contains the identifier and namespace of the OPC UA *ExpandedNodeId* as a string representation. The *TableName* column contains the name of table where archived values of the node are written and read. The *ValueType* and *ValueRank* contain information about the type contained inside the value table and the rank specifies the number of dimensions of the value as specified in the OPC UA standard.

*HistoryValuesVectorDouble* and *HistoryValuesEncoded* tables are used for archiving the spectrum samples and ADI complex data respectively. THe *HistoryValuesEncoded* table contains OPC UA binary encoded data where encoding of the data is specified by *DataTypeNodeId*. This node id is a string representation of the OPC UA *ExpandedNodeId* which is used to determine how binary data is decoded on the server.

Spectra are represented as *YArrayItem* variable types in the ADI information models as shown in previous chapters. This type contains the absorbance measurements in a vector format and the value is archived inside the *HistoryValuesVectorDouble* table. A *YArrayItem* node type contains x-axis information as an OPC UA complex type which contains, for example, the wavelength points of the measurement. This complex type is archived inside the *HistoryValuesEncoded* table.

## 7.2 Data processing system

A chemometric data processing system was implemented as a separate calculation service. It is beneficial to keep the calculation services separate from the actual server that holds the process information as server software should not contain any heavy computations related to chemometric models and data preprocessing. As a result an OPC UA server works only as a data repository in the implemented system and the following figure illustrates some of the data flow interactions between the data acquisition system, calculation software and history database.

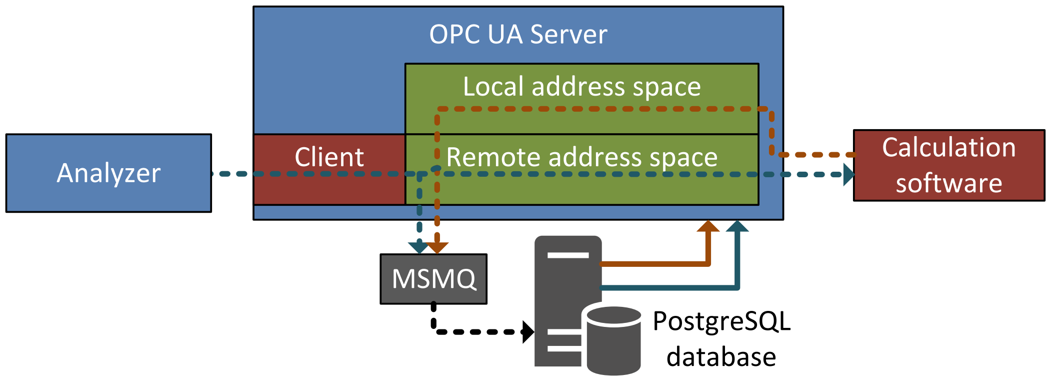
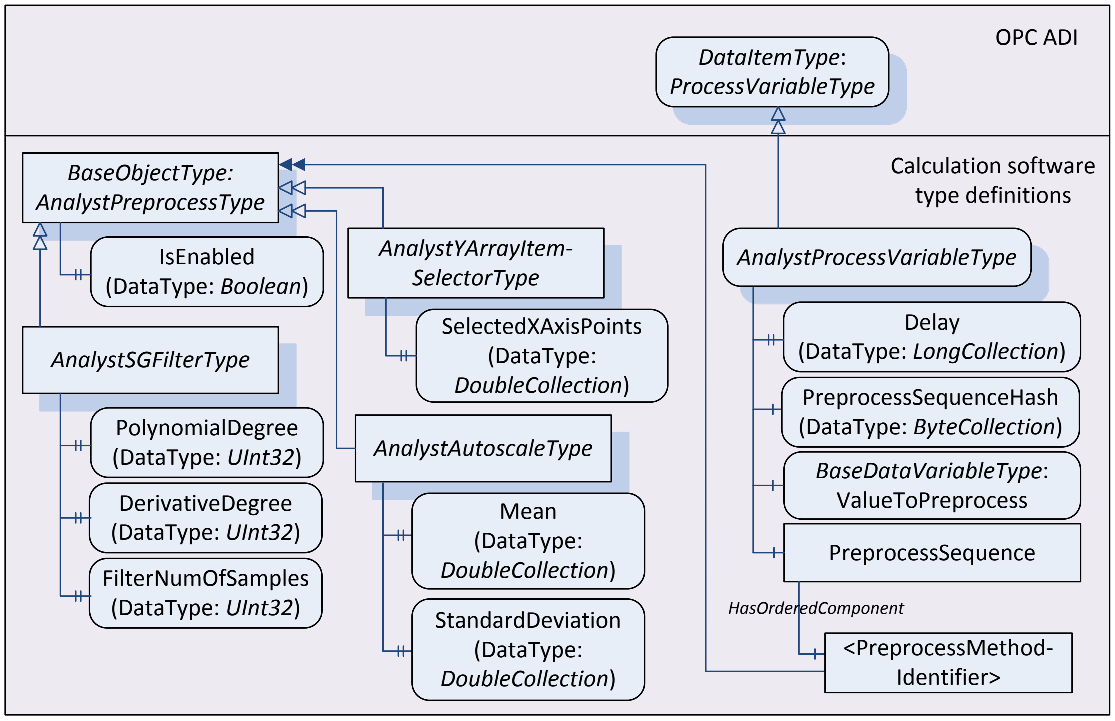


Figure 35. Data flow from the analyzer to the calculation software and finally to the history database

Figure 35 illustrates how an analyzer spectrum measurement is relayed into the history database and to the calculation software through the servers' remote address space. In the illustrated case, the calculation software listens to the value notifications related to the analyzer spectrum measurement nodes it then automatically performs the defined processing sequences with the data and applies the multivariate models. Finally the calculated estimates of the response variables are then written to the OPC UA server. Figure 35 also depicts how the calculated values produced by calculation software can be also archived. In this case the data processing related information models required by the calculation software are located on a local address space of the OPC UA server. These information models contain the configuration and chemometric model specific information. Data flow relations are also modelled using the OPC UA information models and it is highly beneficial to keep all the configuration related information in one place, on the OPC UA server. In this way the system is highly configurable due to the flexibility of the OPC UA information modelling and the calculations can also be configured remotely using any OPC UA compliant software. The following chapters describe the information models used by the calculation software.

### 7.2.1 Data preprocessing information models

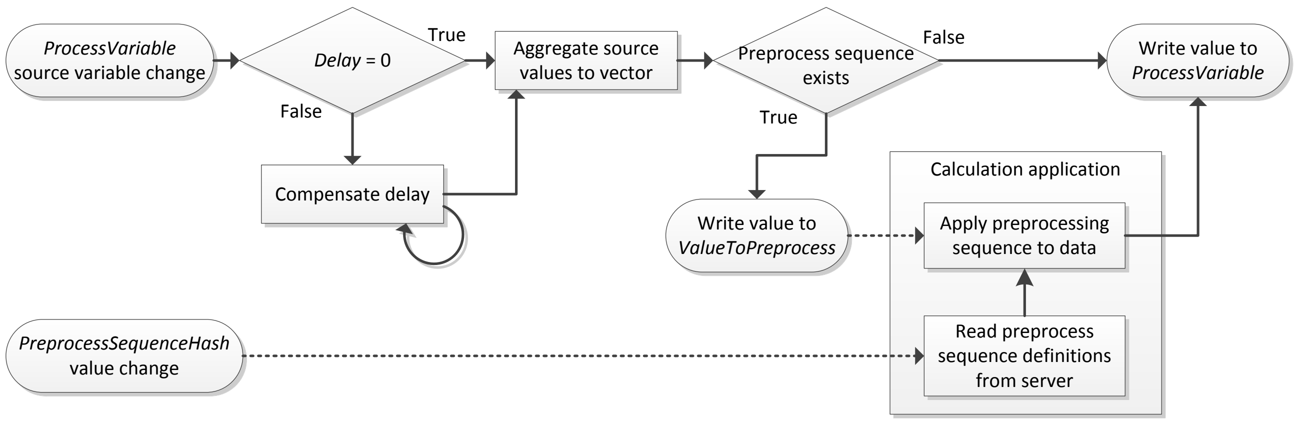
All data preprocessing is performed by the calculation software except for time delay compensation, which is corrected on the server side since delay correction involves buffering of the measurement data. A preprocessing sequence is defined using the OPC UA information models that are utilized by the calculation application. The sequence is first read then applied to the data by calculation software and the following figure describes the preprocessing related information models.



OPC ADI
DataItemType: ProcessVariableType
Calculation software type definitions
BaseObjectType: AnalystPreprocessType
IsEnabled (DataType: Boolean)
AnalystYArrayItem-SelectorType
SelectedXAxisPoints (DataType: DoubleCollection)
AnalystSGFilterType
PolynomialDegree (DataType: UInt32)
DerivativeDegree (DataType: UInt32)
FilterNumOfSamples (DataType: UInt32)
AnalystAutoscaleType
Mean (DataType: DoubleCollection)
StandardDeviation (DataType: DoubleCollection)
AnalystProcessVariableType
Delay (DataType: LongCollection)
PreprocessSequenceHash (DataType: ByteCollection)
BaseDataVariableType: ValueToPreprocess
PreprocessSequence
HasOrderedComponent
<PreprocessMethod-Identifier>

Figure 36. Preprocessing information models used by the calculation software

Figure 36 shows the type definitions for a Savitzky-Golay filter (*AnalystSGFilterType*), an autoscaler (*AnalystAutoscaleType*) and a spectrum wavelength selector (*AnalystYArrayItemSelectorType*) which are sub types of an abstract preprocess method type (*AnalystPreprocessType*). In addition a preprocessing sequence and a time delay are defined inside an extended process variable type (*AnalystProcessVariableType*). This process variable type has a data source defined using *HasDataSource* reference as previously shown. The *AnalystProcessVariableType* supports multiple source variables whose values are aggregated into a vector value.  This vector aggregate is automatically processed by the calculation application and is used to update the *AnalystProcessVariableType* node's value. The formation of a vector aggregate is beneficial since it simplifies the creation of preprocessing definitions, for example, a single preprocessing sequence can be utilized for multiple different data variables. In contrast vector aggregation is not performed when only one data source is defined. The following figure illustrates how data preprocessing logic is implemented on the server and the calculation application.



*ProcessVariable* source variable change
*Delay* = 0
True
False
Compensate delay
Aggregate source values to vector
Preprocess sequence exists
False
True
Write value to *ProcessVariable*
Write value to *ValueToPreprocess*
Calculation application
Apply preprocessing sequence to data
Read preprocess sequence definitions from server
*PreprocessSequenceHash* value change

Figure 37. Preprocessing logic used for the *AnalystProcessVariableType*

Dashed lines illustrate how calculation software monitors the value changes of the server variables. The calculation software updates the final preprocessed value automatically on the *AnalystProcessVariableType* the instance after the defined preprocessing sequence is applied, thus a *ProcessVariable* value always stays in the preprocessed form. The calculation application also refreshes the preprocessing sequence automatically when the *PreprocessSequenceHash* changes. This hash value is an MD5 hash that is formed from the preprocessing sequence and its parameters on the server. A client application can then determine whether preprocessing sequence should be loaded from the server using this value.

### 7.2.2 Chemometric information models

Chemometric information models used by the calculation application extend the abstract ADI chemometric model type. The ADI type is extended into *AnalystChemometricModelType,* which organizes the model parameters and methods under relevant nodes. The extended type contains a variable indicating the model validity and a parameter hash code. The model hash code and the model validity are formed automatically from model parameters and data I/O identifiers with the hash value used by the calculation application to the model changes. The calculation software detects valid models and downloads the relevant matrices used for regression and calculation of the statistical metrics. These matrices are reloaded whenever changes are detected in the valid model. Response variables and statistical metrics are automatically calculated and updated when inputs of the model change.

Two chemometric model types were designed within the work contained it this thesis. These are the PLS and MB-PLS models that were also used in the data modelling part of the experimental part. An overview of the PLS and MB-PLS information models is given in the following figure and tables.

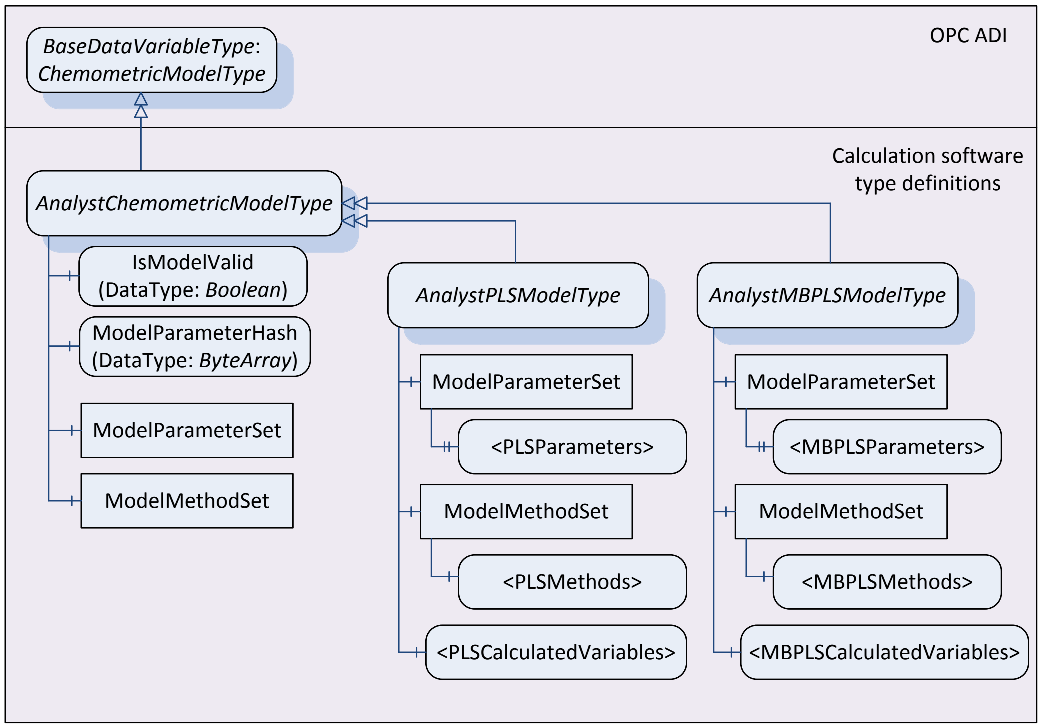


Figure 38. Chemometric information models

*ModelParameterSet* is used to organize all the matrices and other information related to the calculation of the model outputs. The model validity and hash code values are updated automatically from the *ModelParameterSet* properties and model I/O variables and these are determined by comparing matrix parameter dimensions and I/O variable counts. The hash code variable is calculated by first binary serializing the value and then forming the MD5 hash from the raw binary data. Furthermore, the *ModelMethodSet* organizes the methods provided by the models. The following tables show the information models for the concrete chemometric models:

Table 13. AnalystPLSModelType parameters and methods

| **References** | **NodeClass** | **BrowseName** | **DataType** | **ValueRank** | **TypeDefinition** |
|---|---|---|---|---|---|
| ModelParameterSet | | | | | |
| HasProperty | Variable | RegressionMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XScoreMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XLoadingMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XWeightMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | YScoreMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | YLoadingMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XScoreVectorVariances | Double | 1 | PropertyType |
| HasProperty | Variable | SPExMean | Double | -1 | PropertyType |
| HasProperty | Variable | SPExVariance | Double | -1 | PropertyType |
| HasProperty | Variable | YMean | Double | 1 | PropertyType |
| HasProperty | Variable | YVariance | Double | 1 | PropertyType |
| ModelMethodSet | | | | | |
| HasComponent | Method | GetNumLVs | - | - | - |
| HasComponent | Method | GetNumSamples | - | - | - |
| HasComponent | Method | GetNumXvars | - | - | - |
| HasComponent | Method | GetNumYvars | - | - | - |
| HasComponent | Method | AddControlLimitRequest | - | - | - |
| HasComponent | Method | GetControlLimits | - | - | - |
| HasComponent | Method | UploadMatlabModel | - | - | - |

| | | | | | |
|---|---|---|---|---|---|
| Calculated variables | | | | | |
| HasOutput | Variable | <ResponseVariables> | Double | -1 | BaseVariableType |
| HasComponent | Variable | YOutput | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | XScore | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | T2Statistic | Double | -1 | BaseDataVariableType |
| HasComponent | Variable | SPExStatistic | Double | -1 | BaseDataVariableType |
| HasComponent | Variable | ControlLimits | Double | 2 | BaseDataVariableType |

Table 14. AnalystMBPLSModelType parameters and methods

| **References** | **NodeClass** | **BrowseName** | **DataType** | **ValueRank** | **TypeDefinition** |
|---|---|---|---|---|---|
| ModelParameterSet | | | | | |
| HasProperty | Variable | DeflateType | Enum | -1 | PropertyType |
| HasProperty | Variable | BlockNumVars | Int32 | 1 | PropertyType |
| HasProperty | Variable | BlockUsedLVs | Boolean | 2 | PropertyType |
| HasProperty | Variable | XScoreMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XLoadingMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XWeightMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XSuperScoreMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XSuperLoadingMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XSuperWeightMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | YScoreMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | YLoadingMatrix | Double | 2 | PropertyType |
| HasProperty | Variable | XScoreVectorVariances | Double | 2 | PropertyType |
| HasProperty | Variable | SPExMean | Double | 1 | PropertyType |
| HasProperty | Variable | SPExVariance | Double | 1 | PropertyType |
| HasProperty | Variable | YMean | Double | 1 | PropertyType |
| HasProperty | Variable | YVariance | Double | 1 | PropertyType |
| ModelMethodSet | | | | | |
| HasComponent | Method | GetNumBlocks | - | - | - |
| HasComponent | Method | GetNumLVs | - | - | - |

| | | | | | |
|---|---|---|---|---|---|
| HasComponent | Method | GetNumSamples | - | - | - |
| HasComponent | Method | GetNumXvars | - | - | - |
| HasComponent | Method | GetNumYvars | - | - | - |
| HasComponent | Method | AddControlLimitRequest | - | - | - |
| HasComponent | Method | GetControlLimits | - | - | - |
| HasComponent | Method | UploadMatlabModel | - | - | - |
| Calculated variables | | | | | |
| HasOutput | Variable | <ResponseVariables> | Double | -1 | BaseVariableType |
| HasComponent | Variable | YOutput | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | XScore | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | XSuperScore | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | T2Statistic | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | SPExStatistic | Double | 1 | BaseDataVariableType |
| HasComponent | Variable | ControlLimits | Double | 2 | BaseDataVariableType |

The tables show that PLS and MB-PLS information models have some similarities, but are still fundamentally different in terms of how the regression works as well as in the amount of information produced. For example, an MB-PLS model produces statistical metrics and control limits for each of the blocks separately. Both of the information models contain all the matrices obtained during the modelling phase and these matrices are listed in the PLS and MB-PLS chapters of the thesis.

These information models also contain *XScoreVectorVariances* properties which are used for calculation of the $T^2$-statistic as shown in Equation (77). Variances can be also calculated from the *XScoreMatrix*, however a separate vector containing the variances was needed since the size of an *XScoreMatrix* can be very large. This type of vector allows the calculation application to calculate the $T^2$-statistic without the need to download a large score matrix. *SPExMean* and *SPExVariance* are used for calculating the control limits of the $SPE_x$-statistic are shown in Equation (84). An MB-PLS model produces model statistics separately for each of the data blocks, therefore the number of array dimensions related to the statistical metrics needs to be increased by one when compared to the PLS model.

An MB-PLS information model also contains a variable (*DeflateType*) indicating the deflate method used for the model. These methods are the block score and the super score update methods as described previously in Equations (53) and (55). The information model also contains block division information (*BlockNumVars*) containing a number of variables in each of the blocks. The calculation software first unfolds the input variable data into a vector and then divides the data vector into blocks using the *BlockNumVars* information. *BlockUsedLVs* is a truth table containing information indicating which of the latent variables are used in the blocks. This allows some of the subsequent latent variables to be unused in some of the blocks, for example, an MB-PLS model with only a few process variables and thousands of absorbance variables might require the process variable block to have a lower number of latent variables when compared to the absorbance block.

Both of the chemometric models contain a *ControlLimits* variable with a matrix value. This matrix always contains the confidence limit of the control limits in the first column whilst the rest of the columns contain the actual $T^2$ and $SPE_x$ control limits. Calculation of control limits is a request type of operation where the calculation software computes the control limits and updates the matrix. Undefined placeholder values are used for the control limit values before the calculation application updates the matrix.

### 7.2.3 Information flow modelling

Data flow between the different systems and chemometric models is defined using the OPC UA references. This allows the creation of data flow definitions from the source systems to the final response variables in a standardized and structured way. The ADI specification defines *HasInput*, *HasOutput* and *HasDataSource* references which are utilized for modelling of the information flow. *HasInput* and *HasOutput* references are used to describe chemometric model I/Os, whereas the *HasDataSource* reference defines the data source for *ProcessVariable* types that are also utilized by the implemented data preprocessing system. This chapter illustrates how the data flow is modelled between the integrated server, DCS server and analyzer devices.

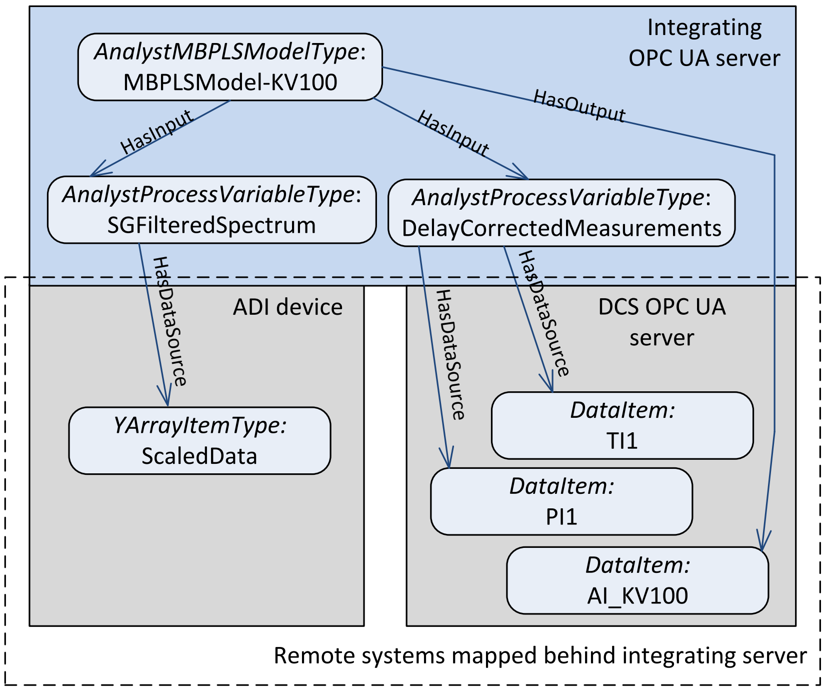

Integrating OPC UA server
AnalystMBPLSModelType: MBPLSModel-KV100
HasInput
HasInput
HasOutput
AnalystProcessVariableType: SGFilteredSpectrum
AnalystProcessVariableType: DelayCorrectedMeasurements
HasDataSource
HasDataSource
HasDataSource
ADI device
DCS OPC UA server
YArrayItemType: ScaledData
DataItem: TI1
DataItem: PI1
DataItem: AI_KV100
Remote systems mapped behind integrating server

Figure 39. An example usage of the chemometric model, data preprocessing and remote data sources

Figure 39 shows an example data flow model for the chemometric estimation of a specific property. The estimation utilizes process data in order to improve the performance of the chemometric model and the example shows how an integrating server is connected to the lower level ADI and DCS servers. These lower level systems contain the analyzer and process measurement data and are mapped on the remote address spaces of the integrating OPC UA server. The data flow models and processing definitions are created on the integrating server where the data source references point to the remote systems. Preprocessed variables (*AnalystProcessVariableType*) are defined to use these data sources for preprocessing. A chemometric model then utilizes the preprocessed data and the final estimate is written to the response variable. Figure 39 also shows how the estimated value can be written to a variable located on the remote DCS server. Services provided by the remote address spaces allow this type of bidirectional data flows between the systems.

## 7.3 Implementation of the information system to the VHVI process

A possible implementation of the created data processing and acquisition system to the VHVI process is presented in this chapter. Figure 40 shows a separate OPC UA network for analyzer devices. This network presents higher amounts of analyzer information to the top level systems when compared to the old milliampere measurement network. An OPC UA analyzer information server works as a single access point for all analyzer information and this server archives specific analyzer data, as defined by the system configurator, to the SQL database. Access point can be utilized by the end users to acquire the plant analyzer data conveniently, through standardized protocols. This is also advantageous since it also allows data integration from other types of sources i.e. process measurements. An integrating OPC UA server was implemented to unite the process measurements and analyzer data sources. A chemometric calculation platform utilizes this data integration by using the analyzer and process measurements for the estimation and monitoring of the specific properties in the VHVI process. Additionally the integrating server contains the chemometric definitions used by the chemometric calculation platform and the results from the chemometric calculations are available on the integrating server. Results of these calculations can be also saved to the history database for further analysis.

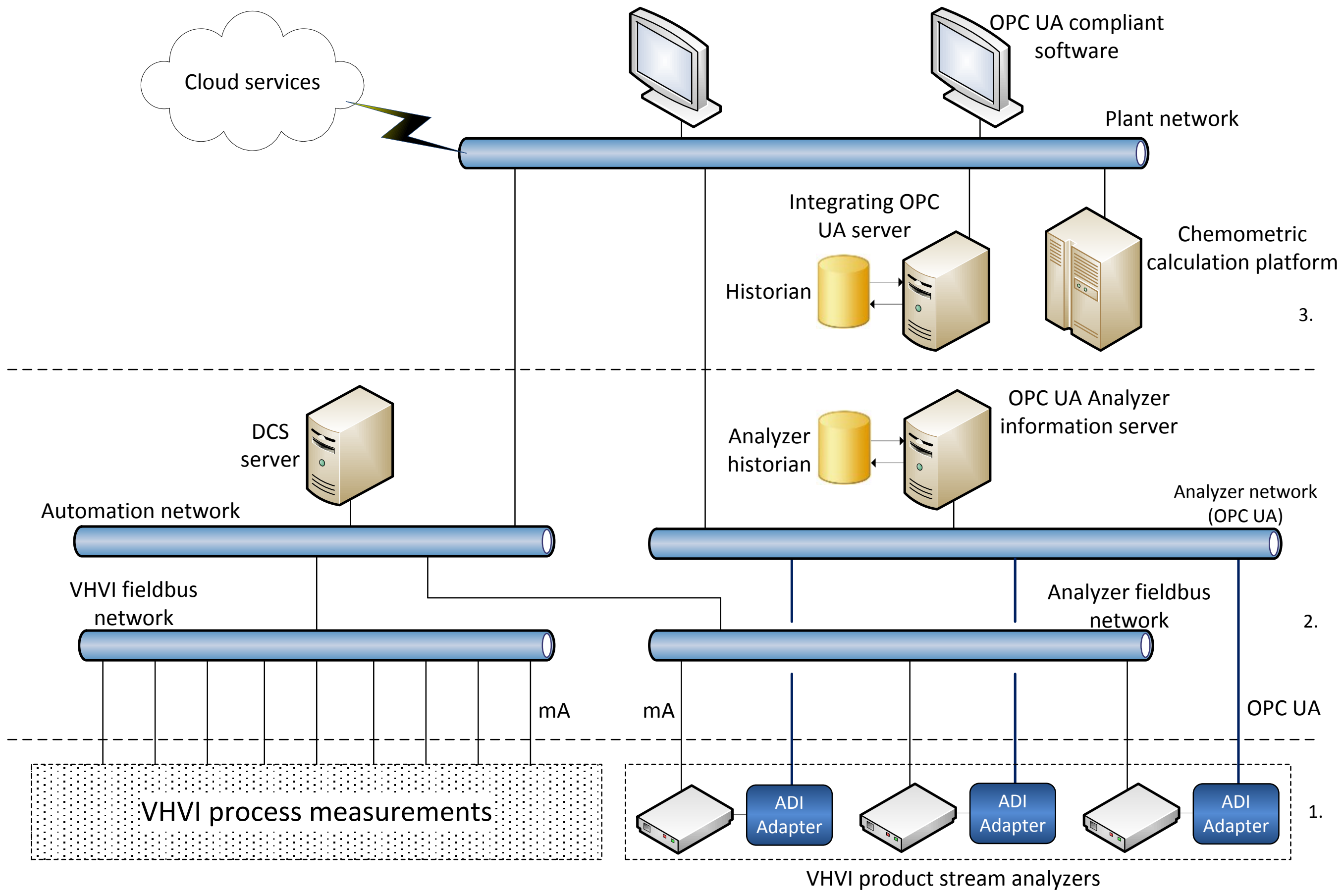

Cloud services
OPC UA compliant software
Plant network
Integrating OPC UA server
Historian
Chemometric calculation platform
3.
DCS server
Automation network
Analyzer historian
OPC UA Analyzer information server
Analyzer network (OPC UA)
VHVI fieldbus network
Analyzer fieldbus network
2.
mA
mA
OPC UA
VHVI process measurements
ADI Adapter
ADI Adapter
ADI Adapter
1.
VHVI product stream analyzers

Figure 40. An example implementation of the data integration system to the VHVI process

This type of data integration has been shown to be beneficial throughout this thesis and the performance of the chemometric models was increased when multiple different types of data sources were utilized. Access to the analyzer information is also more convenient since all the information is readily available to the top level systems. In the old system only the most basic types of analyzer estimates were accessible from the top level systems.

Figure 40 also outlines possible implementation of cloud services which can be used to outsource the chemometric calculations and other data analytics. Data would be transferred to the service provider through the internet using OPC UA protocol. This could be beneficial because the external entity could monitor the performance of the chemometric models and make early corrective actions if the modelling performance starts to degrade. Data confidentiality would require a high level of security measures when the plant data is transferred to the service provider over the internet.

Integration of a laboratory information system (LIMS) to the system is also possible but is not implemented as it is not within the scope of the thesis. This could be beneficial in the future for performance monitoring of the online analyzers, for example, instrument faults and need the for analyzer recalibration can be detected earlier when LIMS data is combined to the data analysis.

Data flow between the different systems is illustrated more clearly in the Figure 41. The dashed gray line in the figure also illustrates how the analyzer results could be transmitted to the DCS server through ADI adapters, superseding the legacy milliampere fieldbus analyzer network. The DCS server can also receive the data analysis results through the integrating OPC UA server.

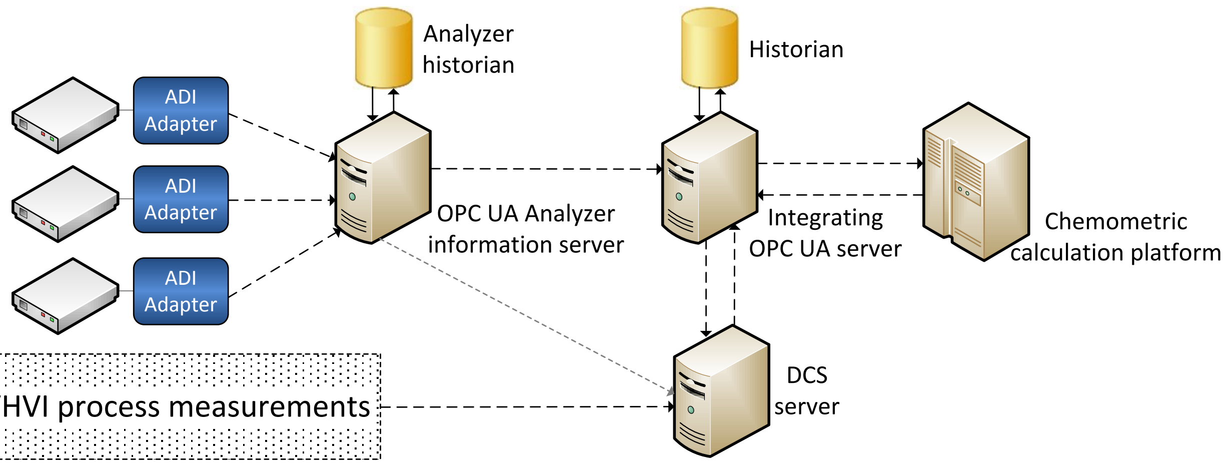

Analyzer historian
Historian
ADI Adapter
ADI Adapter
ADI Adapter
OPC UA Analyzer information server
Integrating OPC UA server
Chemometric calculation platform
DCS server
VHVI process measurements

Figure 41. Data flow between the systems in the VHVI data integration

# 8. Conclusions and further study

Automation systems are becoming more and more integrated through jointly agreed specifications. Interoperability of device communication protocols and information models allows authorized access to the information contained on the devices and the OPC UA standard is a prime example of this mindset where devices can seamlessly communicate with each other. This is beneficial not only for end-users, but also to device manufacturers: customers can choose more freely between different device manufacturers, whilst on the other hand manufacturers can also integrate their devices to existing applications more easily.

Openness also creates new types of applications through the ease of data access and integration. This thesis clearly demonstrates one example of such data integration where all the analyzer information of a process unit becomes readily available to the users and top level automation systems. The created data integration system was fully based on the OPC UA communication protocol and was also utilized on a multivariate calculation platform which was configurable through OPC UA information models. This system refined a combination of plant and analyzer data into more valuable information and it was also shown how the performance of the chemometric product property models was improved with the data integration. The created calculation platform and data acquisition system can also be used in applications where the operational performance of the analyzers and process are monitored through the statistical metrics produced by the chemometric models.

Further studies should examine the possibility of integrating plant laboratory information sources to the proposed system as this could be used to further refine the data and produce more valuable information to the users. For example, laboratory information can be included to the data analysis and used for continuous validation of the analyzer calibrations. The effect of the operational regions of the process should be also studied more carefully when process measurements are incorporated into chemometric analysis. In addition, adaptive chemometric models could be implemented and tested in cases where there is a wide degree of variation within the process. The proposed system makes the implementation of the adaptive models more convenient since the historical data of both the analyzers and process measurements are readily available to the higher level calculation platforms. Moreover the use of the OPC UA allows secure transfer of the plant information through the internet which could be utilized to create cloud services for plant data analytics.

## References

[1] ABB. 2014. Process analytical technology. ABB analyzer brochures. [Online] [Cited: November 13, 2014.] http://www05.abb.com/global/scot/scot267.nsf/veritydisplay/2f340ec8199ebf85c1257cdd0040b9fa/$file/Brochure_PAT_3BHT490777_EN_a_LowRes_140519.pdf.

[2] Yokogawa. 2001. FT-NIR Analyzer NR800. Yokogawa analyzer brochures. [Online] [Cited: November 13, 2014.] http://cdn2.us.yokogawa.com/product_NR800_BU.pdf.

[3] Siesler, H. W. (Ed), et al. 2009. Near-Infrared Spectroscopy. Wiley-VCH, pp. 1-39. ISBN: 3-527-30149-2.

[4] Somoza, Mark. 2006. Graphical depiction of the Morse potential with a harmonic potential for comparison (Image source). [Online] [Cited: August 5, 2014.] http://commons.wikimedia.org/wiki/File:Morse-potential.png.

[5] Burns, D. A. and Ciurczak, E. W. 2008. Handbook of Near-Infrared Analysis Third Edition. CRC Press Taylor & Francis Group, pp. 7-21, 79-231. ISBN: 978-0-8493-7393-0.

[6] Hidajat, K. and Chong, S.M. 2000. Quality characterisation of crude oils by partial least square calibration of NIR spectral profiles. IM Publications, Journal of Near Infrared Spectroscopy, 8 (1), pp. 53–59. DOI: 10.1255/jnirs.264.

[7] Wolf, M., Ferrari, M. and Quaresima, V. 2007. Progress of near-infrared spectroscopy and topography. PubMed, Journal of Biomedical Optics, 12 (6), pp. 062104-062104-14. DOI: 10.1117/1.2804899.

[8] McGlone, V.A., Jordan, R.B. and Martinsen, P.J. 2002. Vis/NIR estimation at harvest of pre- and post-storage quality indices for 'Royal Gala' apple. Elsevier, Postharvest Biology and Technology, 25 (2), pp. 135–144. DOI: 10.1016/S0925-5214(01)00180-6.

[9] Yabe, K., et al. 2011. NIR Spectroscopy of Star-Forming Galaxies at z ∼ 1.4 with Subaru/FMOS: The Mass–Metallicity Relation. Astronomical Society of Japan, Oxford Journals, 64 (3), p. 60. DOI: 10.1093/pasj/64.3.60.

[10] Gans, Peter. 2011. Interferometer for FTIR (Image source). [Online] [Cited: August 5, 2014.] http://commons.wikimedia.org/wiki/File:FTIR_Interferometer.png.

[11] Kohonen, J. 2009. Advanced Chemometric Methods: Applicability on Industrial Data, Ph.D. thesis. Acta Universitatis Lappeenrantaensis, pp. 25-63. ISBN: 978-952-214-814-8.

[12] Rinnan, Å., Berg, F. V. and Engelsen, S. B. 2009. Review of the most common pre-processing techniques for near-infrared spectra. Elsevier, Trends in Analytical Chemistry, 28 (10), pp. 1201–1222. DOI: 10.1016/j.trac.2009.07.007.

[13] Verboven, S., Hubert, M. and Goos, P. 2011. Robust preprocessing and model selection for spectral data. Wiley, Journal of Chemometrics, 26 (6), pp. 282–289. DOI: 10.1002/cem.2446.

[14] Faber, N. M. 1999. Multivariate Sensitivity for the Interpretation of the Effect of Spectral Pretreatment Methods on Near-Infrared Calibration Model Predictions. Analytical Chemistry, 71 (3), pp. 557–565. DOI: 10.1021/ac980415r.

[15] Ren, S. and Gao, L. 2011. Combining artificial neural networks with data fusion to analyze overlapping spectra of nitroaniline isomers. Elsevier, Chemometrics and Intelligent Laboratory Systems, 107 (2), pp. 276–282. DOI: 10.1016/j.chemolab.2011.04.012.

[16] Baffi, G., Martin, E. B. and Morris, A. J. 1999. Non-linear projection to latent structures revisited. Elsevier, Computers & Chemical Engineering, 23 (9), pp. 1293–1307. DOI: 10.1016/S0098-1354(99)00291-4.

[17] Kramer, R. 1998. Chemometric Techniques for Quantitative Analysis. New York: CRC Press, pp. 99-111, 181-183. ISBN: 978-0824701987.

[18] Levina, E., et al. 2006. Estimating the number of pure chemical components in a mixture by maximum likelihood. Wiley, Journal of Chemometrics, 21 (1-2), pp. 24–34. DOI: 10.1002/cem.1027.

[19] Andersson, M. 2009. A comparison of nine PLS1 algorithms. Wiley, Journal of Chemometrics, 23 (10), pp. 518–529. DOI: 10.1002/cem.1248.

[20] Dayal, B. S. and MacGregor, J. F. 1998. Imporved PLS Algorithms. Wiley, Journal of Chemometrics, 11 (1), pp. 73–85.

[21] Mörtsell, M. and Gulliksson, M. 2001. An overview of some non-linear techniques in Chemometrics. FSCN report, pp. 3-22. ISSN: 1650-5387 2001:6.

[22] Sjöberg, J., Lauwers, L. and Schoukens, J. 2004. Identification of Wiener–Hammerstein models: Two algorithms based on the best split of a linear model applied to the SYSID'09 benchmark problem. Elsevier, Control Engineering Practice, 55 (11), pp. 1119–1125. DOI:10.1016/j.conengprac.2012.07.001.

[23] Zhu, P. and Mei, D.C. 2014. Correlation functions of an autonomous stochastic system with time delays. Elsevier, Physica A: Statistical Mechanics and its Applications, 398 (15), pp. 152–161. DOI: 10.1016/j.physa.2013.12.022.

[24] Westerhuis, J.A. and Coenegracht, P.M.J. 1997. Multivariate modelling of the pharmaceutical two-step process of wet granulation and tableting with multiblock partial least squares. Wiley, Journal of Chemometrics, 11 (5), pp. 379–392. DOI: 10.1002/(SICI)1099-128X(199709/10)11:5<379::AID-CEM482>3.0.CO;2-8.

[25] Westerhuis, J. A., Kourti, T. and MacGregor, J. F. 1998. Analysis of multiblock and hierarchical PCA and PLS models. Journal of Chemometrics, 12 (5), pp. 301–321. DOI: 10.1002/(SICI)1099-128X(199809/10)12:5<301::AID-CEM515>3.0.CO;2-S.

[26] Choi, S. W. and Lee, I-B. 2005. Multiblock PLS-based localized process diagnosis. Elsevier, Journal of Process Control, 15 (3), pp. 295–306. DOI: 10.1016/j.jprocont.2004.06.010.

[27] Kohonen, J., et al. 2007. Multi-block methods in multivariate process control. Wiley, Journal of Chemometrics, 22 (3-4), pp. 281–287. DOI: 10.1002/cem.1120.

[28] Bras, L. P., et al. 2005. Multiblock PLS as an approach to compare and combine NIR and MIR spectra in calibrations of soybean flour. Elsevier, Chemometrics and Intelligent Laboratory Systems, 75 (1), pp. 91–99. DOI: 10.1016/j.chemolab.2004.05.007.

[29] T., Kourti. 2005. Application of latent variable methods to process control and multivariate statistical process control in industry. Wiley, International Journal of Adaptive Control and Signal Processing, 19 (4), pp. 213–246. DOI: 10.1002/acs.859.

[30] Kohonen, J., Reinikainen, S.-P. and Höskuldsson, A. 2008. Block-based approach to modelling of granulated fertilizers' quality. Elsevier, Chemometrics and Intelligent Laboratory Systems, 97 (1), pp. 18–24. DOI: 10.1016/j.chemolab.2008.06.015.

[31] Laxalde, J., et al. 2014. Combining near and mid infrared spectroscopy for heavy oil characterisation. Elsevier, Fuel, 133, pp. 310–316. DOI: 10.1016/j.fuel.2014.05.041.

[32] Lopes, J.A., et al. 2002. Multiblock PLS Analysis of an Industrial Pharmaceutical Process. Wiley, Biotechnology and Bioengineering, 80 (4), pp. 419–427. DOI: 10.1002/bit.10382.

[33] Macías-Hernández, J. J. and Angelov, Plamen. 2007. Soft Sensor for predicting Crude Oil Distillation Side Streams using Evolving Takagi-Sugeno Fuzzy Models. IEEE, IEEE International Conference on Systems, Man and Cybernetics, pp. 3305 - 3310. DOI: 10.1109/ICSMC.2007.4413939.

[34] Qin, S.J. 1998. Recursive PLS algorithms for adaptive data modeling. Elsevier, Computers & Chemical Engineering, 22 (4-5), pp. 503–514. DOI: 10.1016/S0098-1354(97)00262-7.

[35] Chen, M., Khare, S. and Huang, B. 2014. A unified recursive just-in-time approach with industrial near infrared spectroscopy application. Elsevier, Chemometrics and Intelligent Laboratory Systems, 135, pp. 133-140. DOI: 10.1016/j.chemolab.2014.04.007.

[36] Li, W., et al. 2000. Recursive PCA for adaptive process monitoring. Elsevier, Journal of Process Control, 10 (5), pp. 471–486. DOI: 10.1016/S0959-1524(00)00022-6.

[37] Dayal, B. S. and MacGregor, J. F. 1996. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. Elsevier, Journal of Process Control, 7 (3), pp. 169–179. DOI: 10.1016/S0959-1524(97)80001-7.

[38] Angelov, P., Xydeas, C. and Filev, D. 2004. On-line Identification of MIMO Evolving Takagi Sugeno Fuzzy Models. IEEE, IEEE International Conference on Fuzzy Systems, 1, pp. 55-60. DOI: 10.1109/FUZZY.2004.1375687.

[39] Plamen, A. and Xiaowei, Z. 2006. Evolving Fuzzy Systems from Data Streams in Real-Time. IEEE, International Symposium on Evolving Fuzzy Systems, pp. 29-35. DOI: 10.1109/ISEFS.2006.251157.

[40] Middleton, R.H. 1988. Design issues in adaptive control. IEEE Transactions, Automatic Control, 33 (1), pp. 50-58. DOI: 10.1109/9.360.

[41] Keithley, R.B., Carelli, R.M. and Wightman, R.M. 2011. Rank estimation and the multivariate analysis of in vivo fast-scan cyclic voltammetric data. PMC, Analytical Chemistry, 82 (13), pp. 5541–5551. DOI: 10.1021/ac100413t.

[42] Eigenvector Research. September 21, 2013. Using Cross-Validation. Eigenvector wiki. [Online] [Cited: October 21, 2014.] http://wiki.eigenvector.com/index.php?title=Using_Cross-Validation.

[43] Mehmood, T., et al. 2012. A review of variable selection methods in Partial Least Squares Regression. Elsevier, Chemometrics and Intelligent Laboratory Systems, 118 (62-69), pp. 62–69. DOI: 10.1016/j.chemolab.2012.07.010.

[44] Centner, V., et al. 1996. Elimination of uninformative variables for multivariate calibration. American Chemical Society, Analytical Chemistry, 68 (21), pp. 3851–3858. DOI: 10.1021/ac960321m.

[45] E.F., Ildiko. 1987. Intermediate least squares regression method. Elsevier, Chemometrics and Intelligent Laboratory Systems, 1 (3), pp. 233–242. DOI: 10.1016/0169-7439(87)80067-9.

[46] MacGregor, J. F. and Kourtl, T. 1995. Statistical Process Control of Multivariate Processes. Elsevier Science Ltd, Control Engineering Practice, 3 (3), pp. 403–414. DOI: 10.1016/0967-0661(95)00014-L.

[47] Paez, J. C. 2007. New Methods Based on the Projection to Latent Structures for Monitoring, Prediction and Optimization of Batch Processes, Ph.D. thesis. Valencia: Technical University of Valencia, pp. 15-20.

[48] Yin, S., et al. 2011. Study on modifications of PLS approach for process monitoring. IFAC, Proceedings of the 18th IFAC World Congress, 2011, 18 (1), pp. 12389-12394. DOI: 10.3182/20110828-6-IT-1002.02876.

[49] Zhou, D. and Gang, L. 2009. Total Projection to Latent Structures for Process Monitoring. Wiley, AIChE Journal, 56 (1), pp. 168–178. DOI: 10.1002/aic.11977.

[50] OPC Foundation. 2010. IEC 62541. OPC Unified Architecture. [Standard] International Electrotechnical Commission

[51] Mahnke, W., Leitner, S-H. and Damm, M. 2009. OPC Unified Architecture. Springer-Verlag, pp. 1-84. ISBN: 978-3-540-68898-3.

[52] Hollender, M. 2010. Collaborative Process Automation Systems. International Society of Automation, pp. 86-98. ISBN: 978-1-936007-10-3.

[53] OPC Foundation, Inc. 2013. OPC Unified Architecture for Analyser Devices Companion Specification Release 1.1. OPC Foundation, Inc.

[54] Yokogawa Electric Corporation. 2007. AMADAS brochure. Yokogawa web site. [Online] [Cited: October 27, 2014.] http://www.yokogawa.com/sg/aps/pdf/BU53H01A02-01E_001_pod.pdf.

[55] ABB. 2013. VN2300 VistaNET 2.0 . Integrated process analyzer network architecture. [Online] [Cited: October 27, 2014.] http://www05.abb.com/global/scot/scot205.nsf/veritydisplay/91a7d4c435e0cec6c1257b72005e72ca/$file/DS_VN2300_VistaNET-EN_A.pdf.

[56] Unified Automation. UaGateway: OPC Server Configuration. Unified Automation documentation web site. [Online] [Cited: October 31, 2014.] http://documentation.unified-automation.com/uagateway/1.3.4/html/configtool_serverconfig.html.

[57] OPC Foundation, Inc. 2013. OPC Unified Architecture for Devices Companion Specification Release 1.01. OPC Foundation, Inc.

# APPENDIX

## Appendix 1. Descriptions of the selected VHVI process measurements

Table 15. Selected VHVI dewaxing reactor measurements

| **Measurement description** | **Unit** | **Calculated** |
|---|---|---|
| Bed 1 inflow temperature average | °C | x |
| Bed 1 inflow temperature difference | °C | x |
| Bed 1 outflow temperature average | °C | x |
| Bed 1 outflow temperature difference | °C | x |
| Bed 1 temperature rise | °C | x |
| Bed 2 inflow temperature average | °C | x |
| Bed 2 inflow temperature difference | °C | x |
| Bed 2 outflow temperature average | °C | x |
| Bed 2 outflow temperature difference | °C | x |
| Bed 2 temperature rise | °C | x |
| Bed 3 inflow temperature average | °C | x |
| Bed 3 inflow temperature difference | °C | x |
| Bed 3 outflow temperature average | °C | x |
| Bed 3 outflow temperature difference | °C | x |
| Bed 3 temperature rise | °C | x |
| Bed 4 inflow temperature average | °C | x |
| Bed 4 inflow temperature difference | °C | x |
| Bed 4 outflow temperature average | °C | x |
| Bed 4 outflow temperature difference | °C | x |
| Bed 4 temperature rise | °C | x |
| Bed temperatures weighted averages | °C | x |
| Bed 2 and bed 1 temperature difference | °C | x |
| Bed 3 and bed 2 temperature difference | °C | x |
| Bed 4 and bed 3 temperature difference | °C | x |
| Reactor inflow temperature target | °C | x |
| Bed 2 inflow temperature target | °C | x |
| Bed 3 inflow temperature target | °C | x |
| Bed 4 inflow temperature target | °C | x |
| Average of inflow and bed inflows | °C | x |
| Hydrogen inflow to bed 2 | kg/h | |
| Hydrogen inflow to bed 4 | kg/h | |
| Reactor inflow temperature | °C | |

| | | |
|---|---|---|
| Bed 2 inflow temperature | °C | |
| Bed 3 inflow temperature | °C | |
| Bed 4 inflow temperature | °C | |
| Hydrogen inflow temp. to the  bed 2 | °C | |
| Hydrogen inflow temp. to the bed 3 | °C | |
| Hydrogen inflow temp. to the bed 4 | °C | |
| Reactor inlet surface temperature | °C | |
| Reactor outlet surface temperature | °C | |
| Reactor inflow | t/h | |
| Recycle hydrogen to the reactor inflow | t/h | |
| Recycle hydrogen pressure | MPa | |
| Reactor inflow pressure | MPa | |
| Bed 1 pressure difference | kPa | |
| Reactor pressure difference | kPa | |
| Reactor temperature difference | °C | |
| Reactor outflow pressure | MPa | |
| Reactor outflow temperature | °C | |
| Total recycle hydrogen flow | t/h | |
| Recycle hydrogen temperature | °C | |

Table 16. Selected VHVI hydrofinishing reactor measurements

| **Measurement description** | **Unit** | **Calculated** |
|---|---|---|
| Bed 1 inflow temperature average | °C | x |
| Bed 1 inflow temperature difference | °C | x |
| Bed 1 outflow temperature average | °C | x |
| Bed 1 outflow temperature difference | °C | x |
| Bed 1 temperature rise | °C | x |
| Bed 2 inflow temperature average | °C | x |
| Bed 2 inflow temperature difference | °C | x |
| Bed 2 outflow temperature difference | °C | x |
| Reactor inflow temperature | °C | |
| Reactor outflow temperature | °C | |
| Bed 1 top temperature | °C | |
| Bed 1 bottom temperature | °C | |
| Bed 2 top temperature | °C | |
| Bed 2 bottom temperature | °C | |

|  |  |  |
|---|---|---|
| Hydrogen inflow to reactor | kg/h |  |
| Reactor inflow pressure | MPa |  |
| Reactor outflow pressure | MPa |  |
| Reactor pressure difference | kPa |  |

Table 17. Selected VHVI atmospheric distillation column measurements

| **Measurement description** | **Unit** | **Calculated** |
|---|---|---|
| Column overhead outflow temperature | °C |  |
| Column reflux flow | t/h |  |
| Column middle distillate outflow temp. | °C |  |
| Column steam inflow | t/h |  |
| Column bottom outflow temperature | °C |  |
| Column inflow temperature | °C |  |
| Column tray 16 pressure | kPa |  |
| Distillation section pressure difference | kPa |  |
| Column top pressure | kPa |  |
| Column tray 10 temperature | °C |  |
| Column tray 16 temperature | °C |  |
| Column bottom liquid level | % |  |
| Column inflow | t/h |  |
| Flash drum liquid level | % |  |
| Reboiler outflow temperature | °C |  |
| Column reflux flow temperature | °C |  |
| Bottom stream filter pressure diff. | kPa |  |
| Overhead settling drum pressure | kPa |  |
| Atmospheric dist. filter return / feed | - | x |
| Atmospheric dist. steam / feed ratio | - | x |

Table 18. Selected VHVI vacuum distillation column measurements

| **Measurement description** | **Unit** | **Calculated** |
|---|---|---|
| Column bottom reflux flow | t/h |  |
| Reboiler inflow temperature | °C |  |
| Column flash zone temperature | °C |  |
| Column top pressure | Pa |  |
| Column top reflux flow from scrubbing | t/h |  |

| | | |
|---|---|---|
| Column lower reflux flow from scrubbing | t/h | |
| Ejector inflow pressure | kPa | |
| Column fuel oil outflow temp. | °C | |
| Column target reflux flow temp. to bed 1 | °C | x |
| Column overhead flow temperature | °C | |
| Column reflux flow temp. to bed 1 | °C | |
| Ejector inflow temperature | °C | |
| Column overhead settling drum temp. | °C | |
| Column stripping gasses inflow temp. | °C | |
| Column higher reflux flow | t/h | |
| Column bed 1 and bottom pressure diff. | kPa | |
| Column bed 4 and bottom pressure diff. | kPa | |
| Column bed 5 and bottom pressure diff. | kPa | |
| Column inflow temperature | °C | |
| Column inflow | t/h | |
| Column fuel oil tray level | % | |
| Column process oil tray level | % | |
| Column bottom level | % | |
| Column bed 5 pressure | kPa | |
| Column bed 4 temperature 1 | °C | |
| Column bed 4 temperature 2 | °C | |
| Column bed 2 temperature 1 | °C | |
| Column bed 2 temperature 2 | °C | |
| Process oil outflow temp. | °C | |
| Process oil outflow | t/h | |
| Process oil separator bottom level | % | |
| Process oil separator reflux flow | t/h | |
| Vacuum dist. reboiler hot oil flow | t/h | |
| Vacuum dist. lower reflux flow temp. | °C | |
| Process oil separator bottom temp. diff. | - | x |
| Fuel oil temp after cooler | °C | |
| Fuel oil flow to storage | t/h | |
| Column mid distillate flow to scrubber | t/h | |
| Process oil separator hot oil flow | t/h | |

## Appendix 2. Box plots of the PLS1 model metrics

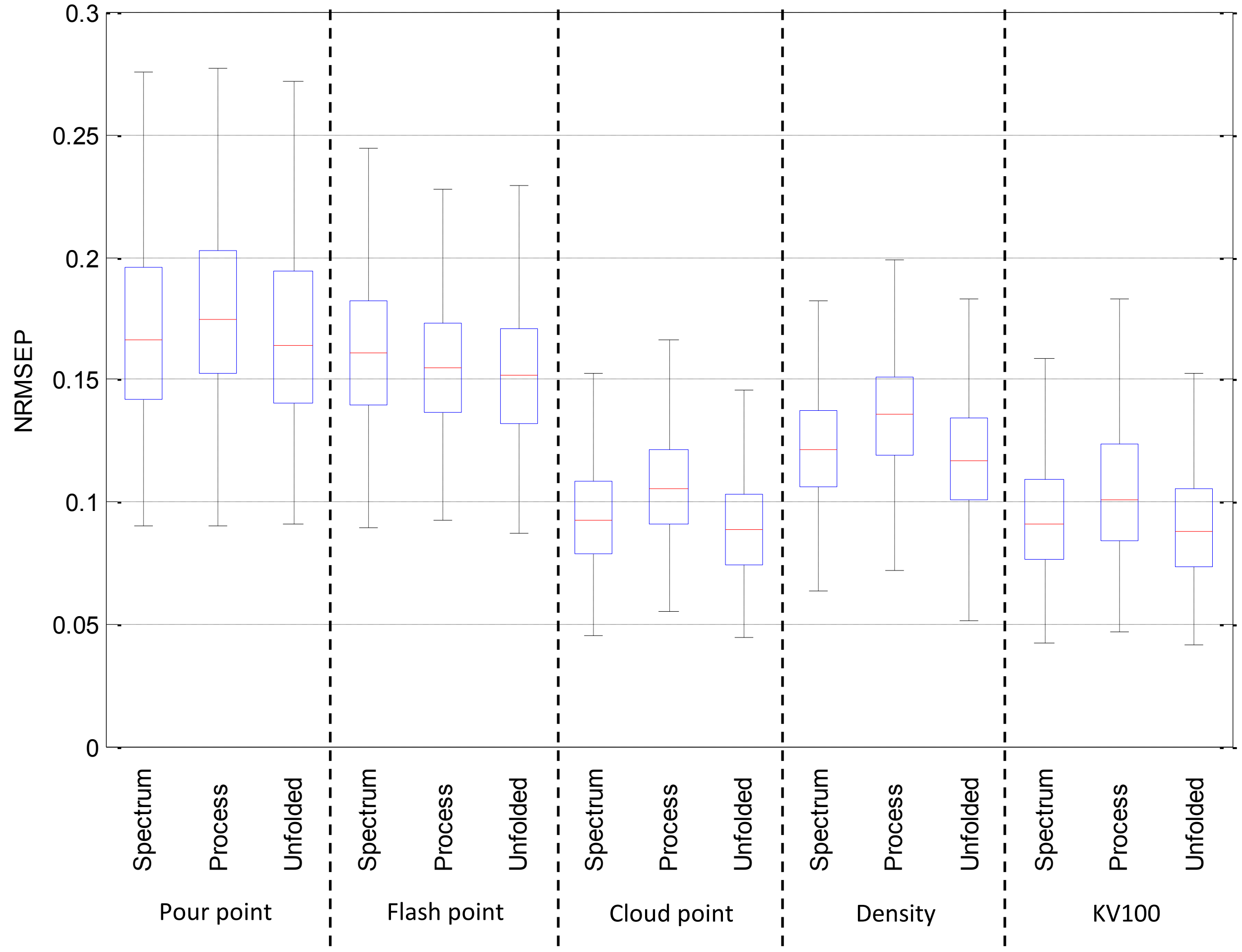


Figure 42. Box plot of the *NRMSEP* metrics for the PLS1 models where a redline is median, box edges represent the 25th and 75th percentiles and dashed whiskers extend to extreme values

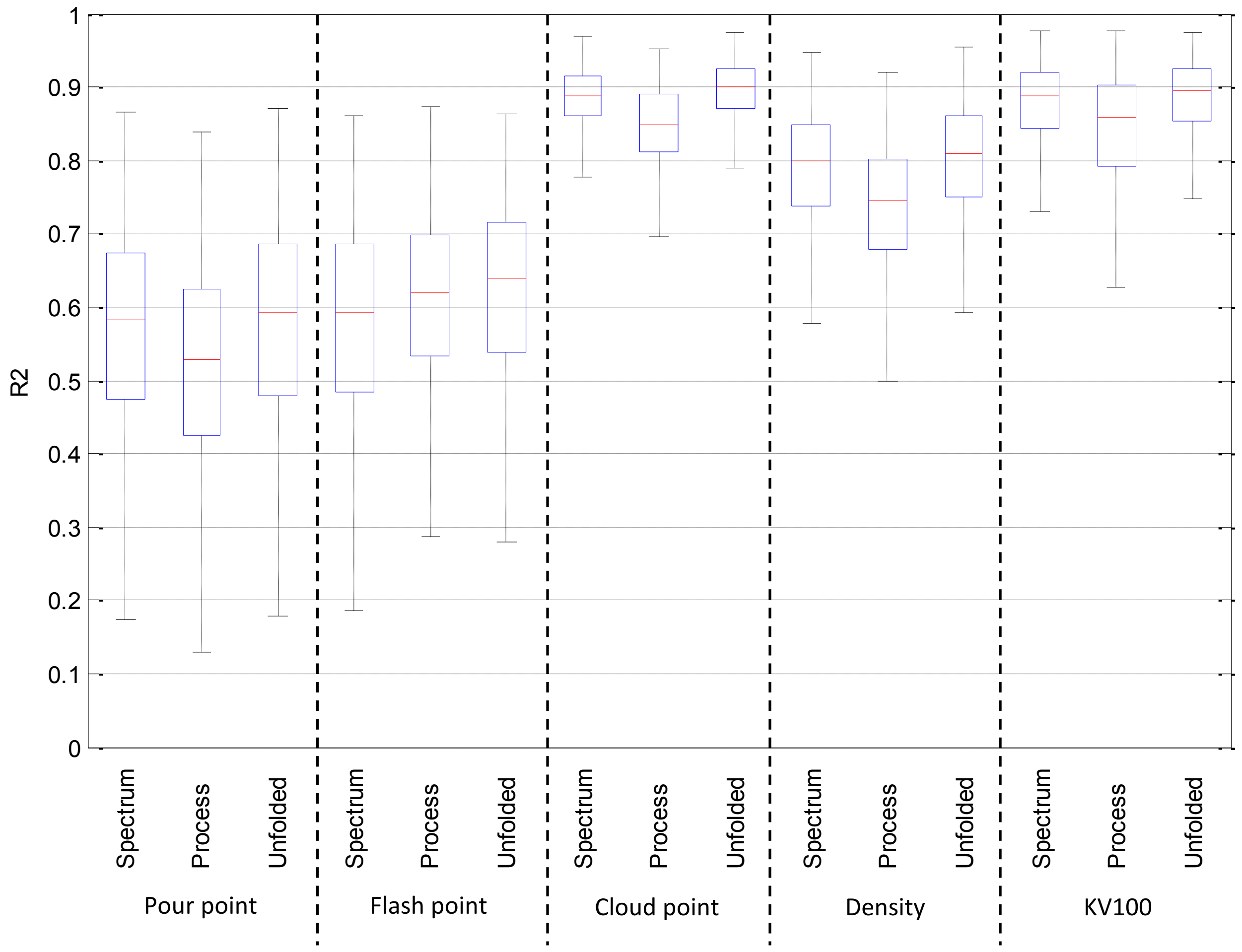


Figure 43. Box plot of the $R^2$ metrics for the PLS1 models where a redline is median, box edges represent the 25th and 75th percentiles and dashed whiskers extend to extreme values

## Appendix 3. Box plots of the MB-PLS1 model metrics

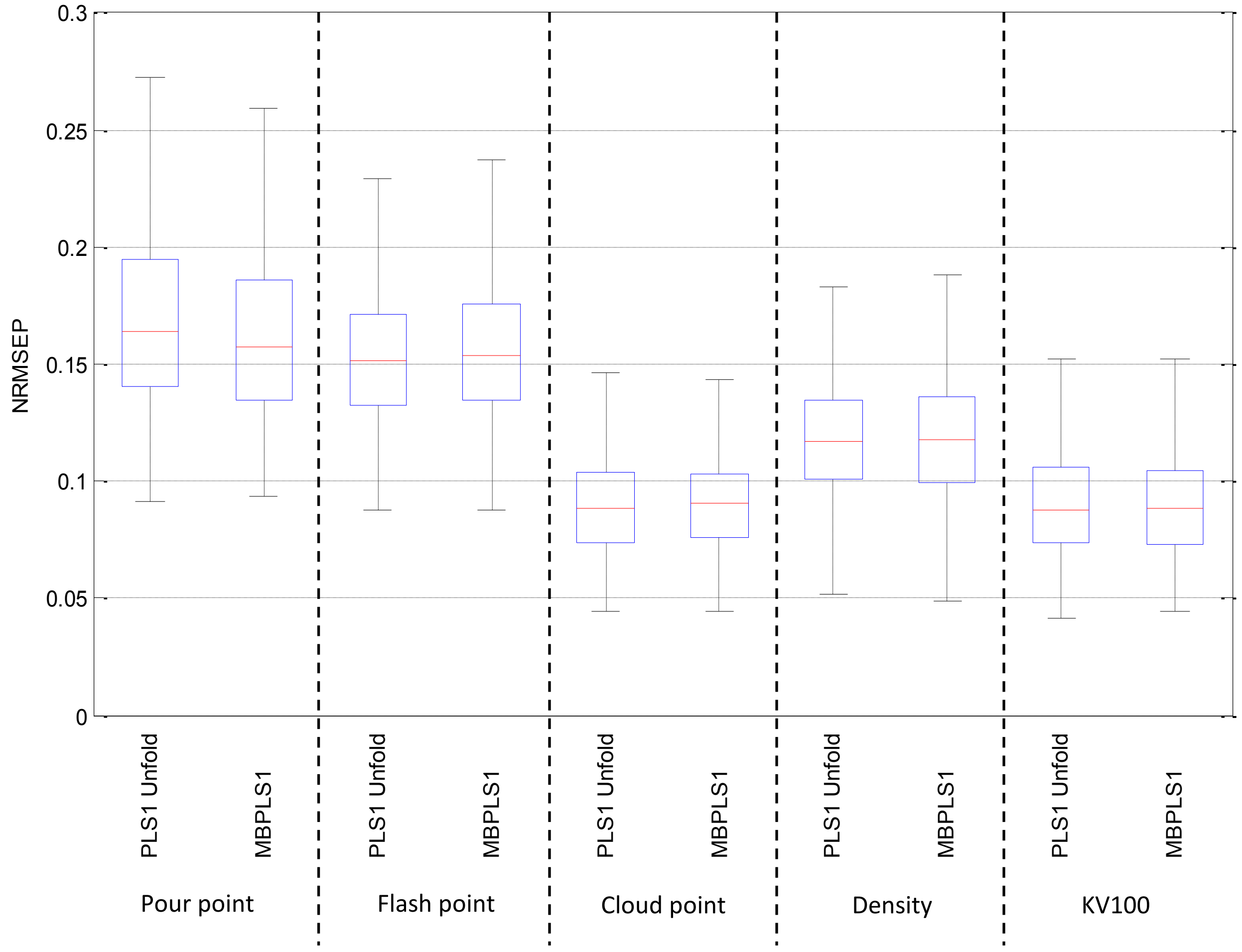


Figure 44. Box plot of the *NRMSEP* metrics for the PLS1 and MB-PLS1 models where a redline is median, box edges represent the 25th and 75th percentiles and dashed whiskers extend to extreme values

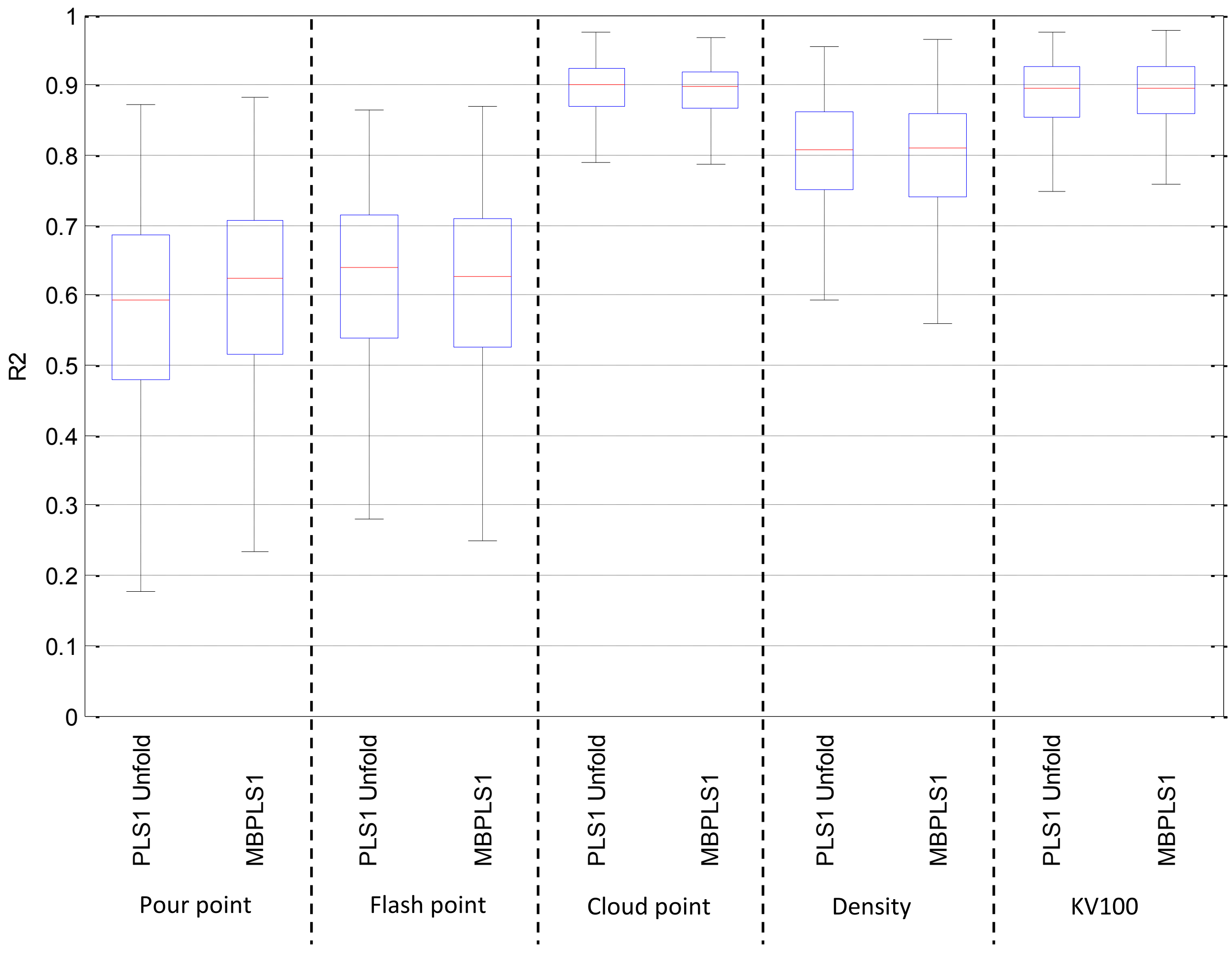


Figure 45. Box plot of the $R^2$ metrics for the PLS1 and MB-PLS1 models where a redline is median, box edges represent the 25th and 75th percentiles and dashed whiskers extend to extreme values

# Appendix 4. Representation of the remote address space services

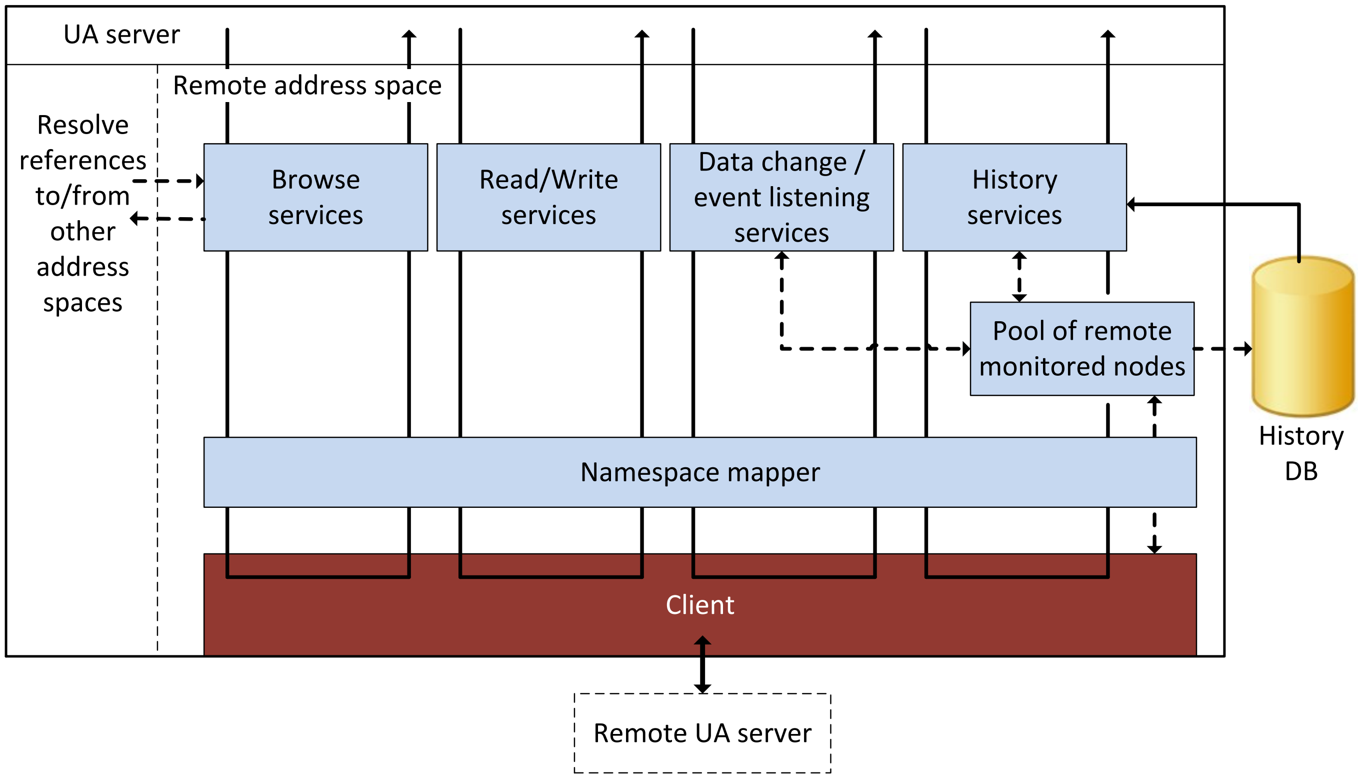


Figure 46. Abstract representation of the remote address space related services