TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

KALLE VIKKULA

IMPLEMENTATION OF POWER BOILER PERFORMANCE CALCULATIONS IN A CLOUD SERVICE

Master of Science Thesis

Examiner: prof. Jukka Konttinen
Examiner and topic approved on
3 January 2018

TIIVISTELMÄ

KALLE VIKKULA: Voimalaitoskattilan suorituskykylaskennan toteuttaminen pilvipalvelussa
Tampereen teknillinen yliopisto
Diplomityö, 82 sivua
Lokakuu 2018
Ympäristö- ja energiatekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Voimalaitos- ja polttotekniikka
Tarkastaja: professori Jukka Konttinen

Avainsanat: teollinen internet, tunnusluku, pilvilaskenta, big data, DCS

”Tieto on valtaa.” -Francis Bacon, 1597

Teollinen internet on yksi esineiden internetin (engl. Internet of Things, IoT) ilmenemismuoto, jonka tavoitteena on digitalisoida teollisuutta. Teolliset prosessit tuottavat suuria määriä tietoa, joka tähän asti on pysynyt vain laitosten paikallisverkoissa. Datan esiin tuominen ja tarkka analysointi tuo uusia mahdollisuuksia niin laitosten omistajille kuin kattilatoimittajillekin. Tämän prosessin keskiössä on keskitettyjen pilvipalveluiden nopea kehitys. Pilvessä dataa voi varastoida lähes määrättömästi ja usealta laitokselta samanaikaisesti, laskentakapasiteetti on suurta, ja hinnatkin ovat nykyään kohtuulliset.

Tämä diplomityö tutki voimalaitoskattilan suorituskykylaskennan toteuttamista pilvipalvelussa. Työssä tutkittiin, dokumentoitiin ja toteutettiin järjestelmä, joka pystyi automaattisesti laskemaan kattilalle tunnuslukuja laitokselta otetun raa’an prosessidatan perusteella. Pääasiallinen tavoite oli tehdä järjestelmästä yleispätevä niin, että se pystyisi mukautumaan minkä tahansa pilveen liitetyn laitoksen dataan.

Työn varsinainen tuote on Python-kielellä toteutettu laskenta-algoritmi, sekä sen ympärille rakennettu logiikka. Algoritmi pyörii pilvipalvelualustalla, ja se voidaan konfiguroida laskemaan automaattisesti minkä tahansa pilveen liitetyn laitoksen tunnuslukuja. Sivutuotteina työ tuotti dokumentaation siitä, miten yksittäinen laitos voitaisiin liittää pilveen; määrittelyn noin 15 erilaiselle kattilan toimintaa kuvaavalle tunnusluvulle; sekä huomioita siitä, millaisia ongelmia digitalisaatioon pyrkivä yritys voi kohdata.

ABSTRACT

KALLE VIKKULA: Implementation of Power Boiler Performance Calculations in a Cloud Service
Tampere University of Technology
Master of Science Thesis, 82 pages
October 2018
Master's Degree Programme in Environmental & Energy Technology
Major: Power Plant & Combustion Technology
Examiner: Professor Jukka Konttinen

Keywords: Industrial Internet, Key Performance Indicator, Cloud Computing, Big Data, DCS

"Knowledge is power." -Francis Bacon, 1597

The Industrial Internet is a subset of the Internet of Things, IoT for short, aiming to digitalize the industrial sector. Industrial processes generate vast amounts of data, which has previously only existed on the local plant computer network. By bringing the data to daylight and performing advanced analytics on it, new opportunities can be discovered. In the center of this process is the emergence of cloud computing infrastructure, allowing for a huge localized data storage and efficient calculation algorithms with an acceptable price.

This thesis studied the procedures and implementation of a system for calculating performance indicators for power boilers. The system was implemented in a cloud-based platform, and it calculated the results from raw process data extracted from the site. The principal aim was to generalize the calculation of these performance indicators in the cloud so that the same calculation procedures could be applied to any plant added to the database in the future.

The main product of this thesis is a calculation script written in Python. The script runs on the cloud platform and can be configured to automatically perform calculations for any site connected to the database. The side products are the documentation of an engineering procedure for adding the data of new plants to the cloud; defining a set of roughly 15 different KPIs applicable to power boilers; and finding possible obstacles a company attempting to leap into the Industrial Internet market can face.

PREFACE

This thesis was done as a part of an Industrial Internet project by Valmet Technologies Oy, Energy R&D department, while officially working for Tampere University of Technology. I would very first like to thank my instructor and TUT side supervisor Tero Joronen, who provided the subject for the thesis and with whom I worked for most of the project. He provided valuable input concerning the scope of the work and supported me during hardships. Next, I would like to thank my examiner Jukka Konttinen for providing academic counsel and keeping my work within reasonable scope. In general, I also thank the Tampere University of Technology and Valmet Technologies Oy for providing this opportunity to work under them.

I would also like to extend my thanks for the following people: Jaani Silvennoinen, Joni Maunula, Tuomas Petänen, Janne Koivuniemi, Tiina Stenvik and Jussi Lautala (for general support within the VII project); Jesse Salmi and Julius Elfving (for support with the data pipeline and databases); Tuukka Harmaala and Sofia Koivumäki (for IT support and counseling); Petri Köykkä, Aleksi Tornberg and Risto Toivonen (for DCS support); Pekka Lehtonen, Ville Ylä-Outinen and Tero Luomaharju (for support with boiler-related questions); Lari-Matti Kuvaja (for support with ES-related questions); Taneli Mutikainen and Salla Tiikko (for COMOS and plant schematics support); Atte Nopanen, Kimmo Djupsjöbacka, Antti Nissinen and Pasi Virtanen (for various VII-related questions); and the Solita and Solutive teams (for support with cloud computing). I also want to present my special thanks to Jukka Sassi, who helped me during this thesis, and directed me to this career in the first place. I would not be here without him.

Finally, I would like to thank my family, friends and loved ones for their continued support before, during and after this thesis.

Tampere, 30.8.2018

Kalle Vikkula

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AWS	Amazon Web Services
BFB	Bubbling fluidized bed
BI	Business intelligence
BQL	Birst query language
BU4	Boiler Unit 4, pilot boiler
CFB	Circulating fluidized bed
CON	Control diagram
CSV, .csv	Comma separated values file format
CYMIC	Valmet CFB product name
DCS	Distributed control system
DE	Drive end vibration measurement
DNA	Distributed Network of Applications, Valmet DCS product name
FbCAD	Function block Computer Aided Design, an automation engineering tool
FGR	Flue gas recirculation
FIC	Flow indicator with controller
GWP	Global warming potential
HFO	Heavy fuel oil
HHV	Higher heating value
HIC	Valve position indicator with controller
IaaS	Infrastructure as a service
IAPWS-97	Standard for calculating the thermodynamic properties of water, by the International Association for the Properties of Water and Steam
II	Industrial Internet
IIoT	Industrial Internet of Things
INFO	Valmet plant historian database product name
INI, .ini	Initialization file format
INT	Interlocking diagram
IO	Input / Output
IoT	Internet of Things
IT	Information Technology
JSON, .json	JavaScript object notation file format
KKS	Kraftwerk-Kennzeichen-System (Ger.), Identification System for Power Plants
KPI	Key performance indicator
LHV	Lower heating value
MCR	Maximum continuous rating
MDM	Master Data Management
MDS	Master Data Service
NDE	Non-drive end vibration measurement
ODBC	Open database connectivity
P&ID	Piping and instrumentation diagram
PaaS	Platform as a service
pH	pH scale of acidity
PI	Pressure indicator
PID	Proportional-integral-derivative (controller)
ppb	Parts per billion
RBAC	Role-based access control

RTD	Resistance temperature detector	
S3	Amazon Simple Storage Service	
SaaS	Software as a service	
SFTP	Secure shell file transfer protocol	
SQL	Structured query language	
TI	Temperature indicator	
VFD	Variable frequency drive	
VII	Valmet Industrial Internet	
VPC	Valmet Performance Center	
VPC	Virtual private cloud	
WWW	World Wide Web	
ZIP, .zip	Zipped file format	

c_{H2O}	H_2O content in flue gas	[%]
$c_{O2,dry}$	O_2 content in dry flue gas	[%]
$c_{O2,ref}$	Reference O_2 content in flue gas	[%]
$c_{x,dry}$	Content of species x in dry flue gas	[mg/m^3]
$c_{x,dry,ref}$	Content of species x in dry flue gas, O_2-referenced	[mg/m^3]
$c_{x,dry,ref,N}$	Content of species x in dry flue gas, O_2-referenced, normalized	[mg/Nm^3]
$c_{x,wet}$	Content of species x in wet flue gas	[mg/m^3]
daily sand	Daily sand consumption	[kg]
ΔT	Temperature difference	[°C]
$E_{fuel,day}$	Daily fuel energy consumption	[MWh]
h	Specific enthalpy	[kJ/kg]
Load%	Steam load percentage KPI	[%]
P_i	Power consumption of machine i	[MW]
P_{own}	Own power consumption KPI	[MW_e/MW_{th}]
p_{ref}	Reference pressure	[1.0135 bar]
PAR	Primary air ratio KPI	[%]
sand consumption	Sand consumption KPI	[kg/MWh]
$T_{bed,avg}$	Furnace bed average temperature	[°C]
T_i	Sensor i temperature measurement	[°C]
T_{max}	Highest temperature	[°C]
T_{min}	Lowest temperature	[°C]
T_{ref}	Reference temperature	[273 K]
V_{hop}	Sand hopper volume	[m^3]
x	Transmitter transmit times	[pcs]

Φ_i	Fuel load at time i	[MW]
λ	Excess air coefficient	[-]
ρ_{sand}	Sand bulk density	[kg/m^3]

C	Carbon (atomic)
CO	Carbon monoxide
CO_2	Carbon dioxide
CH_4	Methane
C_nH_{2n}	Hydrocarbon string
$CaCO_3$	Calcium carbonate
CaO	Calcium oxide, quicklime

$CaSO_3$	Calcium sulfite
Cd	Cadmium
$Fe(OH)_3$	Ferric hydroxide
FeO(OH)	Ferric hydroxide
Fe_2O_3	Ferric oxide
H	Hydrogen (atomic)
H_2	Hydrogen (molecular)
H_2O	Water, steam, moisture, humidity
HCl	Hydrochloric acid
Hg	Mercury
N	Nitrogen (atomic)
N_2	Nitrogen (molecular)
N_2O	Dinitrogen oxide, laughing gas
N_{2r}	Raw nitrogen (mixture of N_2, argon and CO_2)
NH_3	Ammonia
NO	Nitrogen oxide
NO_2	Nitrogen dioxide
NaOH	Sodium hydroxide
O	Oxygen (atomic)
O_2	Oxygen (molecular)
OH^-	Hydroxide radical
Pb	Lead
S	Sulfur (atomic)
SO_2	Sulfur dioxide

1. INTRODUCTION

The 21st century has seen a rapid transformation in the way people perceive data. Data can mean any sort of information, but in this context, it especially applies to data in electronic form, be it contact information, shipping logs, dental records or customer history. The emergence of cheap bulk storage, better data transfer bandwidths and ever faster processors has led to the era of *big data*, where massive amounts of data is analyzed to find hidden relationships, dependencies and trends.

Industrial processes produce large amounts of data about the process itself, as well as the surrounding infrastructure. This data is relevant to the operation of the process, but most likely as such has no meaning to people such as managers or CEOs. This is most notably due to two factors: the inability to do anything with the data either due to technical limitations or missing the required knowledge, as well as company unawareness about the potential of the data. Recently, many traditional industries have awoken to the opportunities that data mining and analyzing could bring, and improved technology has allowed data to be extracted remotely and analyzed in a timely fashion.

This thesis is a part of an Industrial Internet project at Valmet Oyj, a power boiler and process automation supplier. The reasons for Valmet wanting to expand to the digital market can be broken down as follows:

- Ability to offer customers increased value for their products
- Ability to extract information about the performance of the delivered products
- Shift in focus from a pure capital hardware business toward a software-oriented service provider

Part of attaining these strategic goals is the focus of this thesis. A project was undertaken to implement a fully functional calculation procedure for boiler performance indicators. These performance indicators would be used for both customer reporting as well as internal product performance assessment. The project had two primary goals: to generalize the engineering procedure for creating such a system, and to see how well the system would perform. The general outline of the project was as follows:

- The selection of data for extraction from a pilot site
- The definition of performance indicators
- Implementation of the data pipeline from the site to the cloud
- Implementation of the calculation procedures
- Visualization and result verification

In chapter 2, the theory and reasoning for the project are explained. It introduces the concept of the Industrial Internet and its application to the energy sector, gives basics about boiler and plant operation and how the data is relevant to the process, and explores cloud computing. Chapter 3 showcases the actual implementation of the system, starting with chapter 3.1, where the data sourcing procedure is documented. Chapter 3.2 reveals the selected performance indicators as well as why and how they should be calculated. Chapters 3.3 and 3.4 deal with the data extraction and visualization, respectively, and finally chapter 3.5 describes the implementation of the calculation system in detail. The results are explored in chapter 4, with angles for both the usefulness and performance of the calculation procedure. Chapter 5 points out possible stumbling blocks a company in energy sector might face when transitioning to the digital world, and the thesis is concluded in chapter 6.

This project is not the first of its kind at Valmet. Rather, it can be seen as a continuation of the project 'Energy KPI Monitoring' by another thesis worker, Atte Nopanen. In his thesis Nopanen built a functioning data pipeline from a plant to the cloud and demonstrated the implementation of dashboards. This thesis aims to further his work by laying the groundwork for the engineering procedures as well as implementing the calculations in the cloud, rather than using on-site data. On a broader scope, the Industrial Internet is a hot topic for research in its applications for data mining, big data analytics and machine learning. These methods can bring forth unforeseen benefits in data which previously just sat on the plant DCS, such as predictive maintenance of components and helping plan the plant production schedule. [1]

2. BASE POINT AND THEORY

The introduction of the Internet in everyday life has been gradual but rapid. When the World Wide Web (WWW) was launched in the early 1990s, fewer than 0.4 % of the World population were active users [2]. The development of communication systems, networking technology and available services has led to faster bandwidths as well as mobile applications. People are provided with real-time newsfeeds, weather reports and business calendars, and can contact almost anyone, anytime, anywhere.

Interesting information is, however, produced by other things than just people. In an era where people are starting to expect connectivity and data availability, real-time information gathering from different devices is a natural next step. Applications already successfully incorporating such philosophy include public transportation monitoring [3], flight monitoring [4] and postage monitoring [5], and smart devices that constantly monitor and analyze their surroundings as well as perform their intended functions are becoming household appliances [6]. The data these devices produce becomes even more valuable when it can be accessed remotely: a consumer might want to check the contents of the fridge while in-store (end-user value), and the manufacturer could monitor several interesting statistics useful in business-level decision making (provider value). The term "Internet of Things," IoT, broadly means the integration of such smart devices in the existing Internet framework [7].

A subset, or rather, an application of the IoT philosophy, is the Industrial Internet of Things, or just Industrial Internet, II. As the name implies, it is designed to be used in industry-level applications, as opposed to consumer products described earlier [8]. Several different sectors stand to profit from Industrial Internet applications, such as manufacturing, mining, transportation, healthcare and energy [9]. This thesis will concentrate on applications in the energy industry as part of Valmet's own Valmet Industrial Internet (VII) project.

2.1 VII Background

Valmet Oyj is a multi-sector corporation dealing with power plant and paper mill technology as well as process automation solutions. The energy segment totaled some 8 % of net sales in 2016 and, according to Valmet, holds a market position of #1-3 [10]. Products under the Energy name include power boilers, such as circulating fluidized bed boilers (CFBs) and bubbling fluidized bed boilers (BFBs), gasifiers, power and heat plants and emission control technologies [11].

Valmet's chief competitors in the energy sector include Andritz and Foster Wheeler on boiler technology side and ABB, Honeywell, Emerson and Siemens on boiler automation

side [12]. Current competitor offering in the IoT market includes ABB Ability running on their Internet of Things, Services and People platform [13, 14], Andritz Metris [15] and Siemens MindSphere, providing their own Plant Data Services [16, 17]. Both the need and possibilities of developing their own Industrial Internet solutions has been recognized at Valmet, and the project is collectively called *Valmet Industrial Internet*, VII. A part of the project is Valmet Performance Center, VPC, dealing with power plant and paper mill analytics, remote monitoring and support [18].

2.2 Combustion Fundamentals

Combustion is a rapid, exothermal oxidation reaction between oxygen and a combusting agent. Combustible materials exist in all three phases, such as coal (solid), crude oil (liquid) and natural gas (gaseous). All three are fossil fuels, formed in the Earth's crust over millennia from dead organic matter under high pressures and temperatures. These three form the backbone of the global primary energy supply, with a total of 87 % of all primary energy consumed in 2016 [19]. Additionally, an increasing amount of primary energy is made from renewable combustibles, such as forest or agricultural residue, municipal waste or even purpose-grown biomass. As such, combustion is and will for a long time remain an important aspect of world energy production despite increasing emphasis on alternate production means. Combustible materials used for energy production are collectively called *fuels*. [20]

Although the composition of different fuels varies wildly, the most common elements found in any combustible fuel are carbon (C), hydrogen (H), oxygen (O), nitrogen (N) and sulfur (S). These elements are bound differently in different fuels, such as hydrocarbons (methane CH_4, oil C_nH_{2n}) or various organic compounds (peat, coal). The combustion reactions and mechanisms for each of these elements are known. When the fuel undergoes complete combustion, the initial set-up of the atoms in molecules does not matter, since the combustion products will always be the same. Oxygen is most often introduced in the surrounding air, a mixture of 20.948 v-% O_2 and 78.084 v-% N_2, with traces of argon and carbon dioxide. The trace elements are often either ignored or accounted in the N_2 as raw nitrogen, N_{2r}, with almost nitrogen-like thermophysical properties. Each mole (mol) of oxygen for combustion is accompanied by roughly 3.77 moles of nitrogen, which goes unreacted. For any fuel containing the five aforementioned elements, the stoichiometric combustion reaction with air is in the form

$$C_uH_vO_wN_xS_y + \left(u + \frac{v}{4} - \frac{w}{2} + y\right)(O_2 + 3.77N_2) \rightarrow uCO_2 + \frac{v}{2}H_2O + ySO_2 + \left[3.77\left(u + \frac{v}{4} - \frac{w}{2} + y\right) + \frac{x}{2}\right]N_2 \quad (1)$$

where u, v, w, x and y are stoichiometric coefficients with values larger or equal to 0. [20] In the fuel C, H and S are combustible, forming CO_2, H_2O and SO_2, respectively; O acts as an oxidizer with the oxygen in the air and is consumed; and N again goes unreacted.

Fuels are burned because combustion reactions are *exothermic*, i.e. they release heat. For gaseous and liquid fuels, the amount of heat released can be deduced analytically via combustion reactions, which give each reaction a unique reaction enthalpy. Solid fuels require experimental tests in a bomb calorimeter to determine the amount of heat released, and the result is in the strictest sense applicable only to the analyzed sample. For all the fuels, the amount of heat produced is referred to as its *heating value*. Heating values can be *higher* (*HHV*) or *lower* (*LHV*) *heating values* depending if the energy bound in the evaporated water can be recovered. Heating value is the single most defining aspect of a fuel and can be expressed in energy per mass [MJ/kg], energy per volume [MJ/m^3] or energy per mole [kJ/mol] basis. For example, the LHV for coal is around 28.7 MJ/kg, for wood 19.5 MJ/kg and for heavy fuel oil 41.3 MJ/kg. [20]

As a chemical process, combustion can be viewed from different angles depending on the type of the fuel. The combustion of gaseous fuels such as methane CH_4 is dominated by reaction kinetics, which depends on the local concentrations of the fuel and O_2 molecules, and the differing reaction rates for all partial reactions. Flames produced by gaseous fuels can be of either *premixed* or *diffuse* type. Premixed flames have both the oxidizing and combusting agents mixed (common in internal combustion engines), while diffuse flames are produced when streams of the two collide in the flame front (such as in a common lighter). [20]

Liquid fuel combustion is dependent on the fuel characteristics, such as viscosity and evaporation tendency, as well as the system setup and resulting droplet size. Fuels which evaporate easily and are sprayed in small (less than 10 micrometers) droplet sizes quickly change into the gaseous phase and burn homogenously. Heavier fuels with larger droplet sizes form an evaporative region before the main flame front. Large droplets might not evaporate completely, instead a solid shell of coke (the *cenosphere*) forms around the droplet core, burning much like a solid coal particle with a heterogenous surface reaction. [20]

Burning of solid fuels can be generalized in three steps: *drying*, *pyrolysis* and *char burning* (or gasification). In the drying phase, the moisture (H_2O) in the fuel evaporates, leaving behind dry fuel solids. Pyrolysis (or *devolatilization*) is the formation and evaporation of volatile organics, such as different hydrocarbons, on the solid particle surface due to surrounding high temperatures. Pyrolysis is not a combustion reaction but a chemical reaction, and can also happen in anaerobic environments, but in a combustion chamber, the pyrolysis products themselves quickly ignite and burn like liquids or gases. The residual solid matter, known as *char*, contains both the remaining solid carbon as well as the unburnable material. The carbon reacts heterogeneously (solid carbon, gaseous O_2) on the particle surface, forming either CO or CO_2 depending on the amount of O_2 available. Alternatively, if the process aims for gasification, the char can react in oxygen-deficient environments with CO_2, H_2O or H_2 to produce CO, H_2 and CH_4 respectively. The

resulting solid matter, known as *ash*, contains trace amount of carbon and the incombustible inorganic material. [20] The scheme for solid fuel combustion is shown in Figure 1.

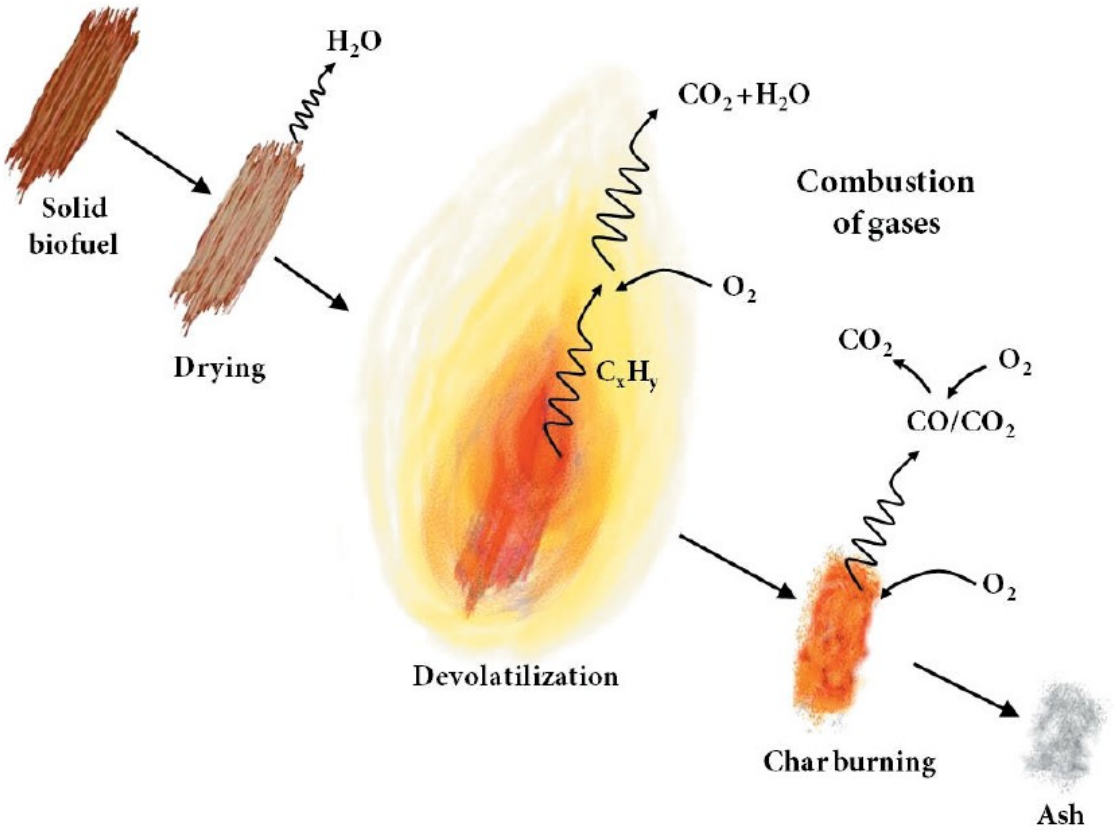


Figure 1. *Steps of solid fuel combustion. [21]*

Besides heat, combustion also generates combustion products. In industrial processes, the combined stream of products in known as *flue gas*. It includes the combustion products of C (CO_2), H (H_2O) and S (SO_2); inert N_2; unreacted O_2; fuel and air moisture; intermediate species formed during the process such as oxides of nitrogen (NO_x); and the inorganic, noncombustible particulate matter such as ash, soot or dust. All but particulate matter are gaseous in form, traveling through the flue gas ducts driven by a fan-induced pressure difference. Ash can remain in the combustion chamber or travel with the gaseous flow to a dedicated cleaning section. Particulate matter is problematic in industrial applications for a few reasons, most notably due to coagulation on heat transfer surfaces. Ash melting point also sets limits to temperature regions. [20]

A typical composition of a coal-fired boiler's flue gas includes about two-thirds nitrogen, 14-18 % CO_2, 10-14 % H_2O, 2-6 % O_2 and <1 % SO_2. Besides these components, the flue gas often includes carbon monoxide (CO), nitrogen oxides (NO, NO_2) and dust, and to a lesser extent ammonia (NH_3), heavy metals (Pb, Hg, Cd) and hydrochloric acid (HCl). Flue gas handling systems aim to minimize the amount of toxic and environmentally harmful species ending in the atmosphere. [22]

CO_2 is the main harmful product of carbon combustion. It is generated via oxidization of carbon:

$$C(s) + O_2(g) \rightarrow CO_2(g) \quad (2)$$

CO_2 is a greenhouse gas and is the baseline for assessing the environmental impact of all the other emission species. The GWP (*Global Warming Potential*) of different substances is measured against 1 unit of CO_2. Greenhouse gases trap solar energy in the Earth's atmosphere, causing the global mean temperature to rise. [23]

CO is a colorless, odorless and toxic gas resulting from incomplete combustion of carbon:

$$C(s) + ½O_2(g) \rightarrow CO(g) \quad (3)$$

High CO emissions are unwanted not only because of the toxicity of CO, but also because a portion of usable chemical energy escapes the combustion process. Formation of CO can be prevented with a high enough excess air coefficient, ensuring complete combustion. [24]

Nitrogen oxides are compounds of nitrogen and oxygen, most notably nitric oxide (NO) and nitrogen dioxide (NO_2) and to a lesser extent dinitrogen oxide (N_2O). Nitrogen oxides are formed during the combustion process in high temperatures from the nitrogen in combustion air (thermal NO_x) and in fuel (fuel NO_x) through the extended Zeldovich mechanism [20]:

$$N_2(g) + O(g) \rightarrow NO(g) + N(g) \quad (4)$$

$$N(g) + O_2(g) \rightarrow NO(g) + O(g) \quad (5)$$

$$N(g) + OH(g) \rightarrow NO(g) + H(g) \quad (6)$$

High flue gas NO_x contents usually indicate problems with either air staging or ammonia injection. NO_x compounds react with sunlight to produce ground-level ozone (O_3) and cause acid deposition and eutrophication in local ecosystems. [24]

SO_2 is a gaseous compound of sulfur released naturally via volcanic activity. Sulphur-rich fuels, such as some varieties of coal, release SO_2 as a combustion product:

$$S(s) + O_2(g) \rightarrow SO_2(g) \quad (7)$$

SO_2 is an air pollutant and a major contributor to acid rain, which causes harmful effects on ecosystems and infrastructure. SO_2 emissions can be reduced by adding limestone ($CaCO_3$) in the furnace, which binds the sulfur in calcium sulfite ($CaSO_3$) via the sequence [24]:

$$CaCO_3(s) \rightarrow CaO(s) + CO_2(g) \quad (8)$$

$$CaO(s) + SO_2(g) \rightarrow CaSO_3(s) \quad (9)$$

Note that trace amounts of additional CO_2 are also generated.

Even though air (or rather, oxygen) is required for combustion, it is not practical to inject all of it from one place in the furnace. In modern boilers air injection systems are divided in sections with dedicated fans, ducts and injection points. Most often these are present as *primary* and *secondary air systems.* Primary air is used as the bulk combustion air during normal operation, and in the case of fluidized bed boilers, acts as both the fluidizing air as well as the main oxidizer for solid fuel combustion. Secondary air systems are connected to various points in the furnace and serve at least two purposes: to provide combustion air for oil or gas burners, and to create pockets of additional air inside the furnace, above the main combustion region. [20]

The reason for this is to create different regions of combustion inside the furnace. Especially in fluidized bed boilers there is a strong draft inside the furnace towards the flue gas ducts, which drives any material entering the furnace outward from the main combustion region, flue gases and fuel particles included. If all the air were injected from a singular point, the fuel particles would escape from the combustion region before achieving complete combustion, leaving unburnable material in the flue gas stream. Secondary air aims to keep the combustion consistent for the complete vertical length of the furnace area, which not only achieves more complete combustion, but also somewhat shifts the resulting flame to the center of the furnace, even if the fuel is injected near the bottom. This also helps to equalize temperature regions inside the furnace. The division of air injection into different phases is called *air staging.* [20]

Air staging is also an efficient method of controlling NO_x emissions. The general idea is to keep the primary combustion region air-deficient to reduce temperatures (less thermal NO_x) and reduce the likelihood of pyrolysis products like ammonia (NH_3) to react with oxygen and produce NO. Staging is then employed in the upper parts of the furnace to create regions of excess air to ensure the completeness of combustion. With this technique a 10 to 50 % reduction in NO_x emissions can be achieved, depending on the fuel quality. However, the purposeful creation of an air-deficient region in the furnace also creates a region where a moderate amount of unburned species (CO, carbon in ash) will form, which needs to be monitored to achieve a balance between NO_x reduction and increase in unburnable materials. [20]

2.3 Boiler Overview

A *boiler* is any device designed to heat a medium, usually water, above the ambient temperature. In industrial applications boilers, also known as *steam generators*, heat water to its boiling point, forcing a phase change [25]. The heat can be generated in many ways, including electric resistors, burning a fuel, concentrating solar rays or by nuclear reactions

[25-27]. As water boils it turns from liquid into steam, which can be heated further in a process called *superheating* [28]. High-pressure, high-temperature steam has a high energy content measured in its *specific enthalpy h* [kJ/kg], which is a common tool for power engineers due to its usefulness in practical applications [28]. Enthalpy is a state function dependent on the temperature and pressure of the medium. Based on the IAPWS-97 definition, a steam flow with a pressure of 100 bar and temperature of 550 °C would have an enthalpy of roughly 3 500 kJ/kg. For reference, water at 1 bar and 25 °C would have an enthalpy of 105 kJ/kg [29].

Boilers introduce outside energy into a process as heat. This energy is in turn converted to other useful forms depending on the process at hand. The steam can be driven though a turbine to generate mechanical energy on the turbine axle, which then is converted into electrical energy in a connected generator [25]. Sometimes there is a need for high-temperature steam in a process, such as paper drying in a paper mill, making the steam valuable as it is [30]. The steam needs to be condensed into water after the turbine, generating waste heat, which can be utilized in district heating applications. It is even possible to design plants to produce more heat at the expense of electrical energy. This is called *co-generation*, and plants running such systems are called CHP (*Combined Heat and Power*) plants. [31]

Power boilers which rely on releasing chemical energy (i.e. burning fuel) can come in a variety of technologies. Valmet has expertise especially in *bubbling bed boiler* technology. A bubbling bed boiler is a boiler with a layer of granular material, such as sand, at the bottom of the furnace (the “bed”) [32]. Below the sand is a grate connected to the combustion air system. When air is blown through the grate it causes the sand to bubble erratically, eventually depicting liquid-like properties at high air speeds; this process is called *fluidization* [33]. The sand itself acts as a heat reservoir in a firing boiler, and the fluidization keeps the temperature profile roughly uniform throughout the furnace [34]. This enables the firing of fuels with variable or low heating values and high moisture contents, such as peat, forest residue, bark, sludge or different types of biomass [35]. Depending on the air speed used, bubbling bed boilers are further categorized in two types, the circulating fluidized bed boilers (CFB) and bubbling fluidized bed boilers (BFB). BFBs use lower air speeds to simply achieve the bubbling effect, while CFBs use higher air speeds, causing the sand to propel along the flue gases to the furnace upper area called the *freeboard* [36]. The sand is separated from the flue gases in a separate, conical chamber called the *cyclone*, where heavier sand particles whirl and gradually fall downwards, allowing the lighter gases to exit at the top. The sand is collected in a component called *loop seal*, which acts as a buffer and typically includes the final superheater in the steam path. From here the sand is returned to the furnace. [37] A depiction of a Valmet CFB boiler (product name “CYMIC”) can be seen in Figure 2.

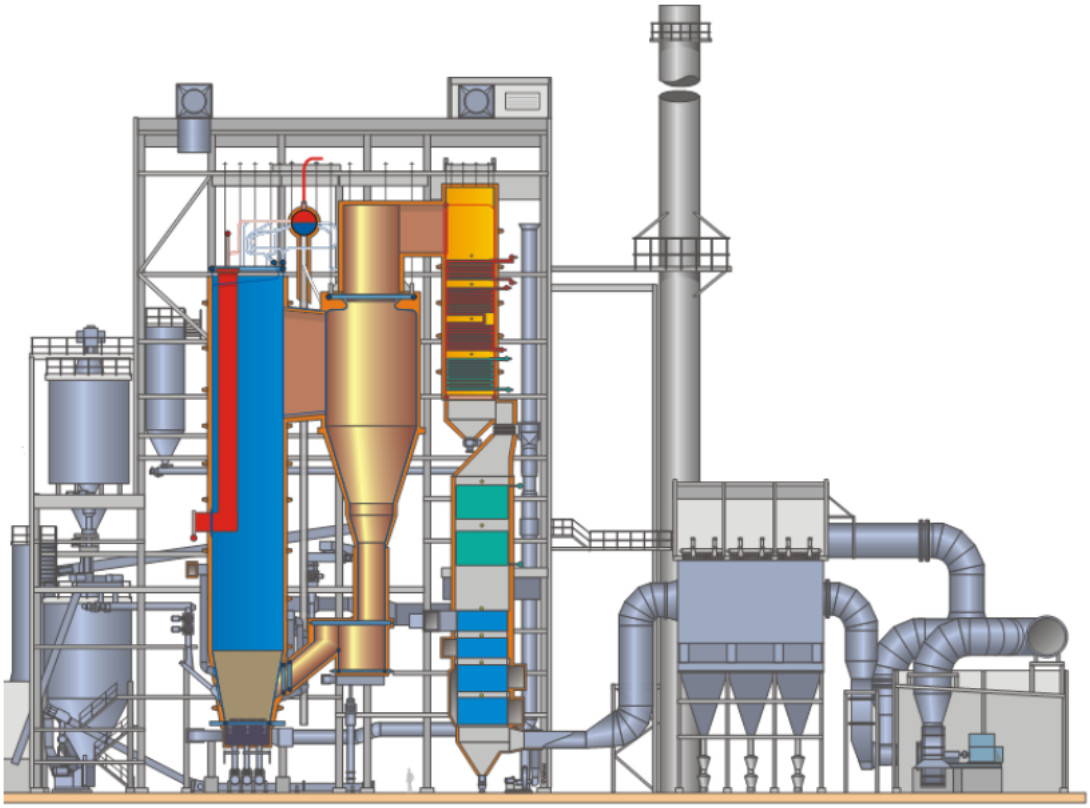


Figure 2. *A Valmet-designed CFB boiler. From left to right: fuel feeding, furnace, cyclone and loop seal, back-pass and flue gas handling. [38]*

Modern power boilers are complex systems, and can be the single most expensive system in a plant [39]. Boiler scope includes not only the water-steam cycle, but also everything pertaining to fuel feeding, combustion monitoring, air flows, flue gas handling and all the auxiliary systems required to maintain an efficient, economical, safe and environmentally sustainable combustion process [25]. At least four distinct subsystems can be found in virtually every boiler:

- Water-Steam system
- Air / Flue Gas system
- Fuel Feeding system
- Auxiliary systems

The Water-Steam system monitors the heating and boiling of water to steam, the primary function of the boiler. Incoming water (called *feedwater*) is first heated to around saturation temperature at the inlet pressure in a component called the *economizer*. Water is then evaporated in tubes running along the walls of the furnace, where the highest temperatures appear, to generate steam. This steam is further heated in superheaters with high-temperature flue gases to reach the nominal temperatures. Steam exiting the boiler is called *main steam*. [40] Besides these key process steps, there is a variety of different subsystems and components depending on the type of the boiler, including a steam drum, reheating system, blowdown and attemporation. [25]

Combustion is basically a rapid, exothermal reaction with oxygen, usually introduced in the surrounding air [41]. In order to achieve a steady combustion, outside oxygen must be constantly supplied to the furnace; this is done via a combustion air system, where combustion air fans pump required amounts of atmospheric air into the furnace. Conversely, most combustible fuels release combustion products in a gaseous form, most notably carbon dioxide (CO_2), water (H_2O), sulphur dioxide (SO_2) and oxides of nitrogen (NO_x). It is also often necessary to supply more air to the furnace than is theoretically required for combustion, leaving excess air in form of O_2 and N_2 in the flue gases. Thus it is practical to group both air and flue gas systems in the same subsystem. Flue gases release heat to the water-steam system as they travel in the flue gas ducts (the *back-pass*), cooling down as they go. After most of the useful heat has been extracted the gases are directed to the flue gas handling system, where the gases are separated from particulates such as dust, ash and soot. The rest is then released at the stack outlet. [42] Emission control is mainly conducted in the air / flue gas system, and includes not just handling the flue gas itself, but also advanced combustion techniques such as air staging [43].

Fuel feeding systems depend greatly on the type of fuels chosen for a plant. Fuels can be in gaseous, liquid or solid form [44] and may or may not require pretreatment such as crushing, heating, pulverizing or slurrying [45-47]. Fuel feeding systems oversee the total amount of fuel entering the furnace and thus total power. Problems in fuel feeding can cause steam parameters to fall to unacceptable levels causing the boiler to trip, and even lead to dangerous situations such as backfires, where fuel still in delivery catches fire [48].

Auxiliary systems include all the supporting equipment and processes necessary for the operation of the boiler. These include the storage and injection of different chemicals like ammonia, phosphates or hydrated lime, water chemistry management and the handling of sand in fluidized bed boilers. Required supporting systems depend on the boiler design. [25]

2.4 Plant Control Systems

As seen in chapter 2.3, the operation of a single modern boiler features numerous different subsystems and processes, each important in function. Ultimately, the boiler is just one part of the total plant, which contains other (some mandatory, some optional) process areas like turbine systems, condensate systems, fuel handling, power substations, refineries and other boiler units [49, 50]. The efficient operation of such large amount of different, co-dependent processes requires some sort of plant-wide monitoring and control system.

The DCS, or Distributed Control System, is such an application. The DCS is ’distributed’ in that the distinct different operation sets are separated from each other both topologically and physically. [51] Operations in the DCS include

- Process monitoring and control via defined automation loops
- Alarm processing and safety routines
- Engineering applications for system management and configuration
- Updating operator displays with real-time system and process data and conveying control orders back to the automation system
- Back-up data storage and upkeep
- History and information gathering

Valmet's DCS is ValmetDNA, short for Distributed Network of Applications [52]. Figure 3 depicts the structure of a Valmet-designed DCS, with most aforementioned operations displayed.

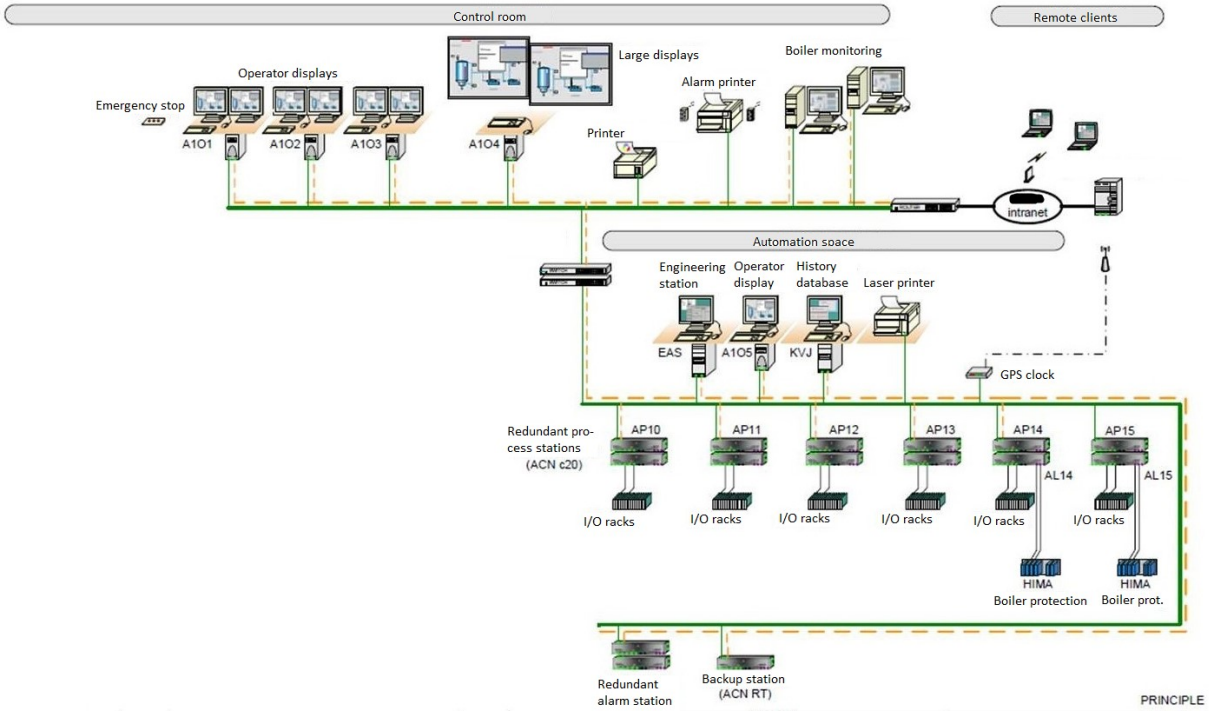


Figure 3. *Topology of a Valmet-designed DCS. [53]*

The basic requirements for process control are:

- Reliable **information** about the state of the process
- The ability to **influence** the process in some way

The first point is handled via different, often redundant measurements in key process points. Measurements can be direct measurements such as those for temperature and pressure, or derived measurements calculated from some other variable, such as boiler power from fuel mass flow. The second point is handled with the necessary equipment integrated in the process, such as pumps, valves, dampers, fans, screw conveyors and hatches. With these the process can be controlled and run in any desired state, either by a human operator or by the logic in automation loops. [54]

Regardless of whether the process is currently run manually or by a computer, there nonetheless needs to be a central control station for human operators. Control is done via operator displays, which contain layers of information depending on the information desired. Overview panels show abstracized versions of process areas for quick analysis while detailed displays show the states of individual components. An example of what an operator can see is depicted in Figure 4.

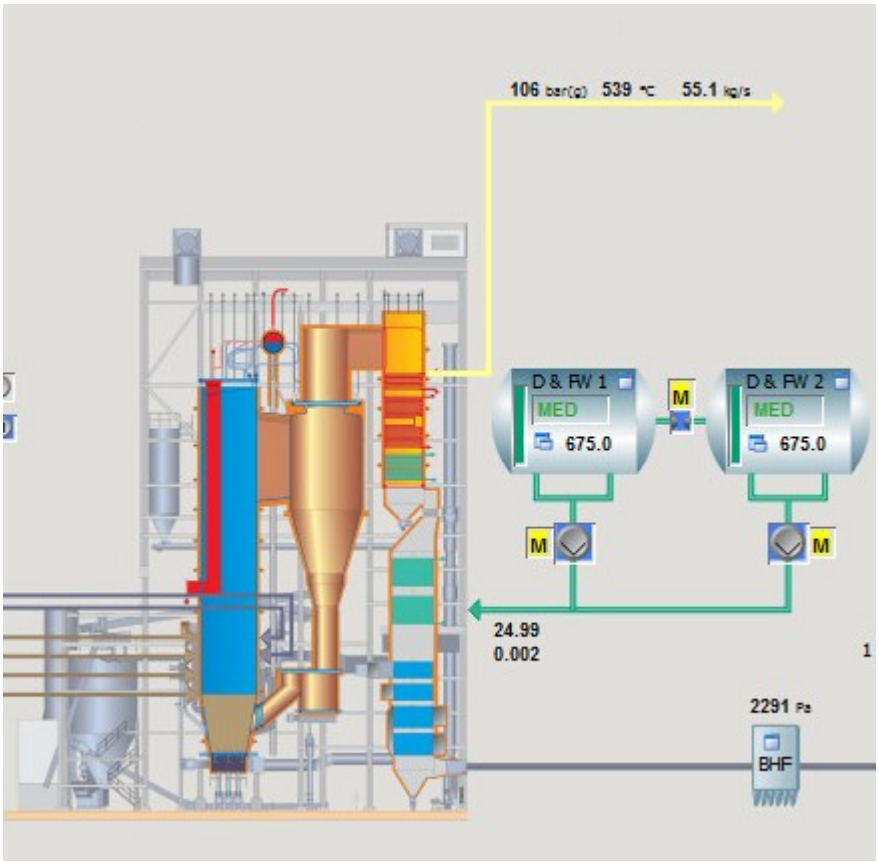


Figure 4. *Part of an overview panel of a CYMIC boiler. The boiler is reduced to a single component, with feedwater and steam parameters included.*

In a modern plant running an automation system all the data exists in electronic form. Measurements might include both local and remote equipment, such as physical pressure gauges along electronic pressure transducers [55], but ultimately for centralized plant operation all process-relevant data must be available at the control room. The significance of this is that, evidently, all the information relevant to the operation of the plant can be accessed via the DCS. Thus far that data has been available only locally on the plant Intranet and only for a limited time [56]. The operators are mostly concerned with the real-time data and some recent trends, but the data contains the potential for a much deeper analysis, appealing to managers and tech suppliers alike.

One goal of VII is to bring this data to daylight by exporting it from the plant and gathering it centrally on a cloud storage. This has several benefits [57]:

- Cloud storage is cheap and easy to scale with the amount of data, cutting the need for large local mass storage
- Dedicated computing environments are capable of much more efficient analysis, cutting the need for local calculation power
- Data from different plants can be analyzed and compared
- Data can be accessed remotely by plant managers as well as tech suppliers

The interval at which the data is collected can be adjusted, with minute-averaged data being a compromise between required storage space and accuracy. Data sent to the cloud includes not only the immediately relevant operational info (e.g. main steam parameters, fuel mass flow, energy produced), but also less tanglible values such as automation parameters. With enough history data these values can be analyzed to construct models of boiler performance, giving new tools to, for example, conduct preventive maintenance on specific components or even devise better ways to run the process and train the operators. [9]

2.5 Cloud Computing

In computing, the *cloud* is a central, remote-accessed database of information and applications [58]. Files on a hard drive or flash drive are only accessible with the computer the drive is plugged in. This is called local storage, or storage tied to a specific device, and moving or copying files between machines simply creates another local storage for that copy. Modifying one copy on one machine will have no effect on copies on other computers. The cloud is the adverse of this: the files are located on a remote database owned and run by a service provider, and the files can be accessed anywhere with an Internet connection and an interface, such as a browser [59]. Cloud services can be simple storage for file transfer and backup [60], e-mail services [61] or even offer cloud applications, allowing working with the same files regardless of the machine used [62]. This is different from local storage where even if you transferred files between machines, you'd also need the necessary software to work with them, such as a spreadsheet program for spreadsheets, and the performance would also be influenced by the computing power on the local machine. Finally, the files in the cloud can be shared with and accessed by members of organizations, groups and communities, with varying read/write permissions. This makes group work smoother and alleviates the problems arising with different-versioned local copies.

Cloud computing can be broken in three levels of abstraction [63]:

- Infrastructure as a Service, IaaS, which provides a low level of abstraction and a highly customizable base for cloud-based applications. The provider supplies scaleable storage, computing and networking services, everything else must be developed by the customer. Examples include Amazon Web Services (AWS) and Microsoft Azure.

- Platform as a Service, PaaS, provides an operating system and some tools to start developing cloud applications. Examples include GE Predix and Siemens MindSphere, both of which are II solutions.
- Software as a Service, SaaS, which has the highest level of abstraction. The service itself is a tool or an application ready to be utilized immediately, rather than having to program it in-house. Examples include Office365 and Dropbox.

Regardless of the service used, all cloud services provide scaleable storage, calculation and networking capabilities. Billing is handled per resource usage, with larger data vaults and more intensive calculations costing more. From a customer point of view this eliminates the need to guess and needlessly oversize local IT hardware, as there is always just enough capacity provided. When there is little need for storage and calculations, the running costs will also be lower. This model of outsourcing hardware and infrastructure also relieves the customer from other associated heavy lifting, such as installing, patching, configuring and server management. [64, 65]

Earlier studies in Valmet's cloud capability and infrastructure have been conducted by Nopanen [57], abridged here. The goal of VII is to make plant data accessible remotely and to enable high-level analytics of the data. For this four things are needed:

- A system to gather and transfer the data
- A storage and management system
- A calculation and publishing system
- An analytics system and tools

The basic infrastructure of the VII Platform can be seen in Figure 5.

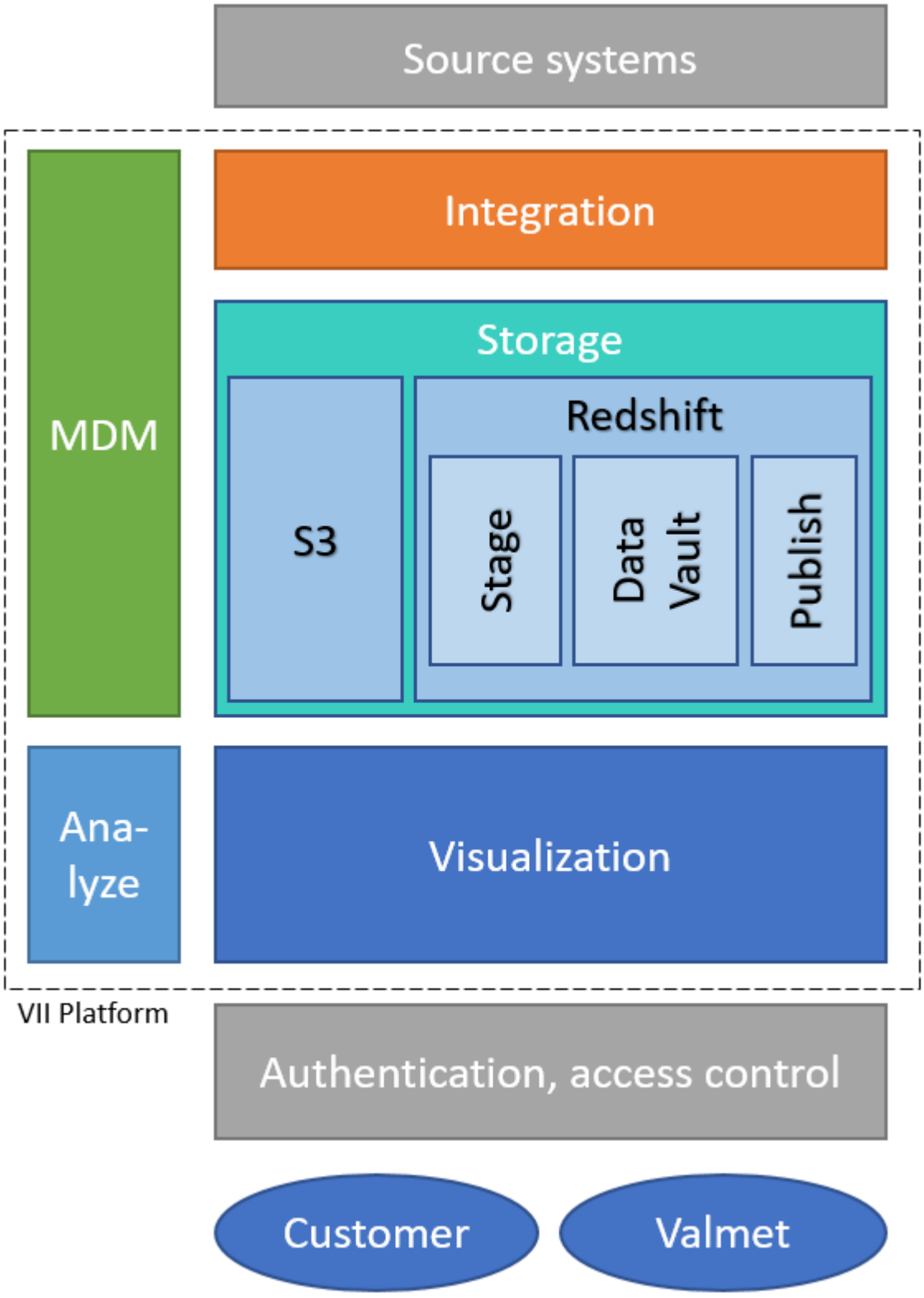


Figure 5. *Basic structure of the VII. Adapted from [57]*

VII data gathering begins at the plant with dedicated scripts, which probe the DCS for relevant continuous data. Data is structured in minute-average format and includes the scan time and date, name of the measurement in the local DCS and the measurement

value. The data is transferred (or *integrated*) to the VII platform using *Secure Shell File Transfer Protocol* (SFTP) and then processed to include additional metadata such as Valmet's internal identification number for that specific plant. In the cloud there are two data containers, the S3 and Redshift. S3 is a bulk type of storage meant to house all incoming data as raw data, while Redshift is a ”dense compute” type of storage with excellent calculation capabilities. The general idea is to only have the most relevant data in Redshift, with S3 providing space for the rarely-needed information. [57]

All data enters through S3 and is additionally sent to Redshift, which has three processing areas. The first is the staging area which determines whether the data should continue on to Redshift and if it should include any additional metadata. Data is then structured in an organized format, or *modeled*, in Redshift Data Vault. Data models describe the relationships between different types of data and tie them together with predefined keys. Data is organized in tables designed to store as pin-point information about data as possible, and the tables have relations called *links* to describe their co-dependencies. For example, there could be tables including information about customers, plants and temperature measurements, and the desired information is extracted from these tables with correctly formulated queries. Queries are made in Structured Query Language, SQL, running on top of the data in data vault. When a query is made the relevant data is probed in the models and necessary calculations are performed (such as getting day-averaged values from minute data), and the results are stored in a publish layer, which the visualization tools can then access. [57]

Visualization is the layer where the customer accesses the data. The tool used is a Business Intelligence (BI) application called *Birst*, which has the necessary visualization features built in. Data is displayed on predefined views called *dashboards*, which are configured to show specific information in a preset format. The SQL queries are integrated in the dashboards during development, and the user can define the timeline from which to get the values from. Dashboard visibility can be restricted on a per-user basis with Role-Based Access Control, RBAC. [57]

The MDM, or Master Data Management, layer is a utility layer for system configuration. It includes features such as maintenance notifications and access management through RBAC, and the Tagmaster, which enables master data configuring for DNA tags. Finally, the Analyze layer contains the analytical calculations as scripts of code such as Python, R or SQL. [57]

In his thesis Nopanen built a functioning system with Kuopion Energia Oy. The end result was a set of dashboards named Energy KPI Monitoring, with things like superheater corrosion and production statistics displayed. A key difference from Nopanen's work is that in Kuopio, the calculation was done on-site and sent readily to the cloud. This thesis aims to bring the calculation to the cloud itself in a generalized form, applicable to any plant with only slight modification. Not only will this fully utilize the cloud computing

philosophy, but all the results for similar plants will also be comparable with each other. [57]

In order to fully deploy VII in both upcoming and already existing plants, the following need to be defined:

- A comprehensive list of what data is needed for a thorough analysis of any type of plant
- What are the interesting performance indicators derived from that data
- How are those indicators calculated
- A generic naming convention for each signal depicting the same thing in different plants, e.g. main steam pressure
- Production costs and customer billing model

2.6 KKS and Tags

KKS is short for *Kraftwerk-Kennzeichen-System*, German for *Identification System for Power Plants*. Power plants feature large amounts of widely different infrastructure and equipment, including pipes, motors, valves, preheaters, circuit breakers, sensors, steam silencers, rotary feeders and storage tanks. For design, operation and maintenance alike it is crucial to have some sort of logical naming convention for all the components and devices to easily identify each one. [66]

KKS defines a set of rules which each component name should conform to. The rules are in the format of breakdown levels, with each level featuring more precise information. A breakdown can be found in Figure 6.

Breakdown level	0	1	2	3
Designation	G	F_0 $F_1F_2F_3$ F_N	A_1A_2 A_N A_3	B_1B_2 B_N
Data character	A or N	N AAA NN	AA NNN A	AA NN

Figure 6. *Breakdown levels in the KKS [67].*

Breakdown level 0 is a code for different units in a plant. It can be either alphabetic or numeric, and is generally somehow bound to the unit in question. For example, a plant with two boiler units could designate boiler 1 as 1, boiler 2 as 2, and common systems for both as 0. [68]

Breakdown level 1 is the process identifier. The first character is a number for identifying identical process areas within the same unit, such as parallel feedwater tanks. Next three letters are process area codes defined in KKS handbooks. In general, the first letter tells

the system category, the second letter the system type and the third letter the subsystem type. The last two numbers are for numbering within the same system, such as for different burner units. [68]

Breakdown level 2 is the equipment unit code. The first two characters are letters defined in handbooks, with the first describing equipment category and the second the equipment type. Next are three numbers for numbering the equipment on the same line. The last character is a letter reserved for fine division, such as the need to designate one valve as A and another as B, if not otherwise applicable via numbering. [68]

Breakdown level 3 is the component code, used in very fine division. One could identify the different electrical components in a piece of equipment, such as separating the motor part from the feedwater pump. The first two letters are KKS defined codes and the last two numbers are for numbering. [68]

For example, 60LAC12AP001 would be the first (001) pumping unit (AP) on the second (12) feedwater pumping line (LAC) which is a unique process (0) for boiler six (6).

The problem with KKS is that it is not adopted globally, meaning it is possible to come across plants with entirely different naming methods. Moreover, the rules set by KKS are in no way exhaustive; there are no set rules for numbering of equipment or processes, for instance. Some companies might modify the scheme for their own needs, such as to include additional breakdown levels for plant identification, and others might drop the breakdown level 3 altogether. [66] Valmet has its own internal rules for applying the KKS, and it's not far-fetched to assume the same goes for other power boiler suppliers. Base line is that Valmet might designate some equipment with one process code and some other company with another, and there is no higher authority on which one would be the absolute correct one.

Since automation deals closely with installed equipment in a plant, and all equipment is already given a name via KKS, it's only logical to name the automation calculation and control procedures (called *loops*) by the component they are most related to. [69] Sometimes it is suitable to separate different types of control over several loops, such as control modules and interlocking logic, requiring additional naming. For instance, the control loop for the aforementioned pump could be the name of the pump, and the interlocking logic loop could have the name 60LAC12AP001L. What is important is that by searching by the equipment code it is possible to find all the loops related to the operation of that piece of equipment.

Automation loops are basically just a sequential collection of different elementary procedures, which are represented as blocks on graphical user interfaces. Basic operations include basic arithmetic, limiting, minimum-maximum selection, logical expressions and lead-lag blocks, among more advanced procedures. There might be different types of basic operations available depending on the environment, but ultimately the function of

a loop comes from the interactivity between the blocks. Blocks have a name, an execution order and input and output ports, which are connected between blocks. The output of one port is the input of another, and the signal value should be the same in both. In the DCS, a *tag* is the name of a single signal. Tags refer to the ports in automation blocks. [70] This is better illustrated in Figure 7, which depicts the Valmet Function block CAD.

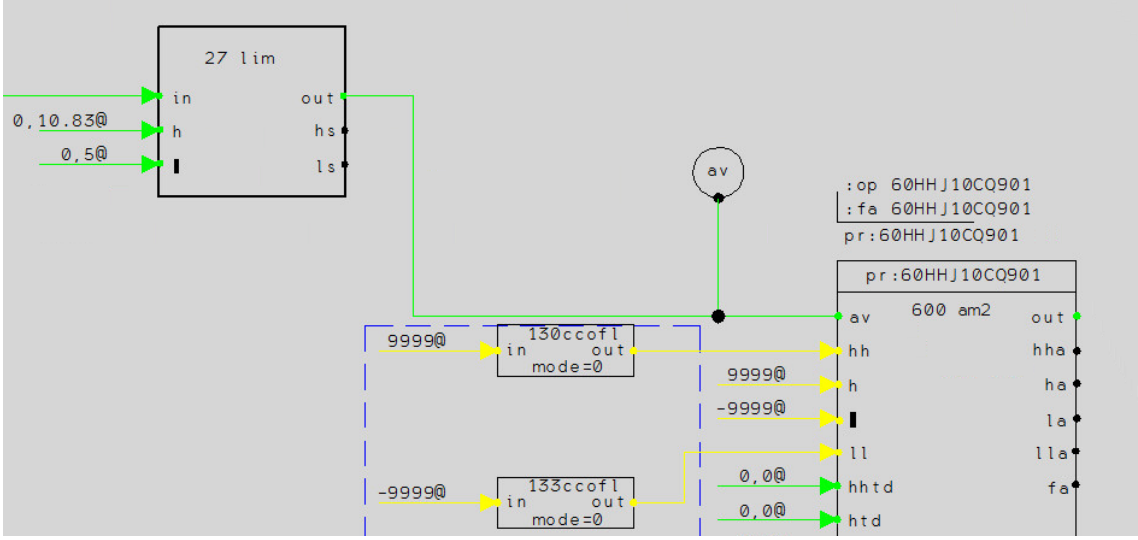


Figure 7. *A section of an automation loop containing a limiter block and a measurement block. The automation tool used is FbCAD. [70]*

In Figure 7, the green arrow depicts an analogous value (i.e. a decimal number) and has some value when arriving to the limiter block's ”in”-port. The block filters the value between 5 and 10.83 and sets the output value of the ”out”-port accordingly. At this point the result value will be in three places at the same time: the limiter block's ”out”-port, the measurement block's ”av”-port (short for *actuator value*) and the reference symbol's input port. (References simply transport values between each identically named reference.) The full names of the ports in the DCS are **pr:60HHJ10CQ901.F:27lim:out** and **pr:60HHJ10CQ901:av** respectively, and both hold the same value. The naming is a bit different because the limiter block needs to be referenced by the loop it's contained in (60HHJ10CQ901) and the fact it's on the sheet (.F), while the measurement block is actually an independent module by the name pr:60HHJ10CQ901, and it could be named differently if wanted. The same logic applies to every block and signal in the automation system. The internal behavior of the blocks is preset, and the automation engineers give them suitable parameters and inputs and make the necessary port connections to construct the desired control system behavior.

As seen, finding an interesting value, such as a specific temperature measurement, in the DCS requires knowledge of the KKS identifier of that temperature sensor. And as said, the KKS naming conventions between plants can vary. This is not really a problem when dealing with single plants, because the naming and numbering conventions will most likely be uniform, but for an efficient Industrial Internet this might cause problems. One

of the goals of VII is to offer Fleet Management solutions featuring comparisons of different plants, and for that the data to be compared should be unambiguous. One part of this thesis was to construct a unified, generic renaming convention that ideally could also be quickly deployed. Since KKS codes do have a rigid structure an algorithm could be utilized, but the results should still be verified by hand.

2.7 KPIs for Boilers

KPIs, or *Key Performance Indicators,* are tools for monitoring the health and status of a business, operation or product, and are employed in many business areas. The relevant KPIs vary between businesses and applications: some are strictly for financial monitoring, while others might track non-monetary values, such as emissions or customer satisfaction. KPIs are tracking and analysis tools for any values the company might want to monitor. [71]

Each KPI must be specified and developed for the current needs of the company. In order to be applicable, a KPI should include and show four points [71]:

- Any prior KPI data
- Target values vs. actual values
- Goals to be fulfilled
- Progress benchmarks

A generalized plan for developing KPIs is illustrated in Figure 8. The plan is from a study of energy efficiency KPIs for manufacturing sector.

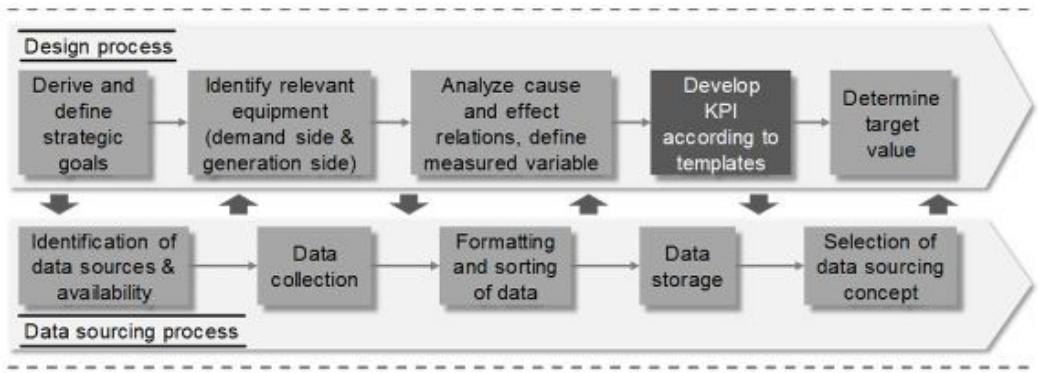


Figure 8. *Suggested process for developing KPIs. [72]*

The plan in Figure 8 has two parallel processes, the design and data sourcing processes, which interact with each other during the development. The design process is centered on the creation of new KPIs via the sequence of required tasks. First, the goals are defined. Then the necessary equipment is identified and measured variables are defined. The actual KPI is then implemented, which can be done with templates or manually, depending on

the KPI at hand. Finally, target values are determined. Simultaneously, there needs to be a plan for the identification, collection, formatting, storage and access of the relevant data in order to evaluate the KPIs. It could be possible that some KPIs, although viable, would require data from a source that is hard to measure or automate, making its application less useful. [72]

Table 1. *A selection of different KPIs applicable to boilers. [73]*

KPI Type	**Description**	**Examples**
Energy	Different forms of energy, e.g. electricity, fuel, heat	Energy Input / Produced Output Energy Output / Energy Input
Material	Raw materials, fuel, water, chemicals	Raw Material Input / Produced Output Waste Deposit / Produced Output
Operation	Operational stability	Operation Time / Time Period Overall Equipment Effectiveness
Control	Control performance	Control Error Variance: Set Point – Measured Value Control loops in manual mode / Total amount of loops
Maintenance	Impact of too low or too high maintenance	Maintenance costs / Produced Output over time period Number of alarms / Time Period
Inventory	Inventory utilization	Throughput Rate / Average Inventory

Equipment	Condition monitoring	Measured – Predicted Performance

For power boilers there are a few categories in which performance could be measured. Accurate indicators about performance in different areas such as raw material usage, maintenance, operation and control, product quality, safety, energy efficiency and scheduling are always welcome. Table 1 presents some suggestions for a CHP plant but is in no way exhaustive; there is no higher authority on what KPIs should be calculated on plants, and no set way how they should be calculated. Each business develops the KPIs based on its own needs and available data. [73]

Finding and recognizing the KPIs is in itself useful, as it can reveal areas of low performance. Low performance is caused by different types of waste, such as energy inefficiency, material waste, increased downtimes or reduced quality, and finding these weak links is the first step in remedying the situation. Many businesses however don't have the appropriate guidelines to track and enhance performance. Lindberg *et al*. [73] provides one method of finding signals in processes that could be tied to the performance of the plant. In this method, the following steps are taken:

1. Selection of signals to log. It is better to include too many rather than too few.
2. Acquisition of historical data, e.g. over 6-12 months, of the selected signals
3. Removal of signals with no standard deviation and signals from when the plant running conditions weren't nominal
4. Calculation of the selected KPI based on the historical data
5. Search of any signals which both **correlate strongly** with the selected KPI and are **possible to adjust**
6. Changing of these signals in the right direction for KPI improvement

The data should be gathered from a period of nominal operation, as disturbances could lead to false assumptions about correlation. Some experiments may also be required to find causalities between data, for example control loop set points and the KPI. [73] In order to recognize the signals that influence the KPI, all signals and combinations of them must be evaluated. For this, equation (10) can be used:

$$F(c) = \sum_{t=1}^{N} [KPI(t) - f(y(t), c)]^2 \qquad (10)$$

Here, *F(c)* indicates the signal correlation (via the *sum of square errors*) and should be calculated for each signal combination. The smaller the value of *F(c)*, the stronger the correlation between the KPI and the selected signal. The process signal itself is the function *y(t)*, applied in the function *f(y(t),c)* given by the user. *f* is a function of the process signal *y(t)* and the scaling parameter *c* over the discrete time period of *t=1* to *t=N*

– that is, the history data at selected points. [73] For example, if f were a linear function of two signals y_a and y_b, the function would be in the form

$$F(c) = \sum_{t=1}^{N}(KPI(t) - [c_1 y_a(t) + c_2 y_b(t) + c_3])^2 \quad (11)$$

Note that F is a function of the scaling parameters c_1, c_2 and c_3, not t. [73]

3. SYSTEM IMPLEMENTATION

This thesis is a part of a larger Industrial Internet project. This chapter describes the overall execution of the project and documents the procedures taken.

3.1 First Steps: Data Selection Procedure

The project aims to streamline Valmet's internal product performance assessment procedures by defining generic performance indicators for the entire product range. Each delivered boiler is more or less unique and tailor-made, but the general working principles don't vary much. All boilers will generate steam and consume fuel, only the technical details see much variation.

The plant chosen for the pilot system is a CHP plant. The boiler in question is a CYMIC-type CFB boiler, and it acts as a replacement for two older coal-fired units. It uses biomass as main fuel, coal as supporting fuel and heavy fuel oil (HFO) as start-up fuel. The plant produces electricity, district heating for the surrounding region and process steam for local industry. The boiler has a thermal power in the range of 300-450 MW_{th}. Designation number for the boiler is 4, making it Boiler Unit 4, or BU4. Both the boiler and plant DCS are supplied by Valmet, making this an excellent baseline project.

The very first thing was to get an overview of the plant and boiler setup. General characteristics, such as steam generation and furnace technicalities, were examined first, and from there more specific systems were targeted. The air preheater is of rotating, regenerative type, and the flue gas is filtered through a baghouse filter system containing multiple compartments. Injected chemicals contain ammonia (for NO_x reduction), limestone (for SO_2 reduction), phosphate (for feedwater quality control) and sodium hydroxide (NaOH, for water pH control). Besides that, there are standard boiler systems such as bottom and fly ash removal, sand feeding, sootblowing and start-up burners.

After the plant and boiler setup technicalities were examined, compiling the data gathering plan began. This was a very iterative process and included the opinion of many experts. A power boiler generates vast amounts of data, a lot of which remains unseen by the operators and exists solely inside the automation loops. Nowhere near all this data is of practical importance with respect to boiler performance, and thus it was necessary to first identify only the relevant information. Even though the cloud can accommodate massive data vaults, logging many unneeded signals causes two things:

- Increased cost of storage
- Longer search times

The following data was suggested for logging by various experts at Valmet:

- All continuous data describing the state of the process, mainly measurements
- Active equipment (i.e. motors) conditions and loads
- Passive equipment (i.e. valves) positions
- Controller performance (setpoints, process measurements, control outputs)
- Derived data calculated in the DCS that is obviously relevant to the process (such as boiler load from fuel mass flow)
- Main interlocking signals for boiler operation
- Signals outside the boiler scope which help answer the question: "why was the boiler run the way it was during that particular time?" Such signals include total electricity generated and total district heating load.

In general, the common wish by the experts was that you could see the cause-and-effect -relationships in the boiler by examining the logged data.

Next step was to identify the individual signals by name. This was by far the most time-intensive phase, since the relevancy and suitability to the above criteria needed to be assessed on a per-signal basis. However, the consensus among the experts was that it is better to log even the signals that might seem quite irrelevant at first, just in case that particular signal happens to be needed at some point. As such, much of the work was balancing between the efficiency (not too much data) and effectivity (enough data for the intended use) of the system. It should be noted, however, that in practice it is quite easy to add additional signals for logging afterwards should the need arise during operation - the point of this step was to compile a first version of a generalized data collection plan, making future work easier.

Since both the boiler and the DCS were supplied by Valmet, it was possible to obtain detailed schematics of both systems. The physical boiler layout was detailed in the piping and instrumentation diagrams (P&ID) and the DCS logic was available on interlocking (INT) and control (CON) diagrams. Additionally, access was received to a copy of the operation system containing the very same operator displays and automation loops as the real plant did, making it possible to delve deep into the inner workings of the automation system. This turned out to be invaluable when determining the types of the data to be logged as well as the actual names of the signals.

The first discretization was done between physical measurements and DCS signals. Measurements tell the process values via the integrated sensory equipment. They transduce the measurement into electrical current, which is then interpreted into numeric digital values at the automation IO (input-output) racks. The DCS values are more abstract, since some exist purely in a digital form as intermediate calculations, Boolean values, fault indications, timestamps etc. For example, no sensor which could measure the heating value of a solid fuel on an inserter screw accurately and in real time exists, so the fuel load is

calculated from the screw speed, capacity and either the latest sampled heating value or an approximated constant. Other significant DCS signals include interlockings, which are purely digital (binary values) by nature.

The P&IDs only show the scope of the Valmet delivery, hiding any instruments provided by other suppliers or subcontractors. Valmet scope included the following:

- Water-steam cycle from the feedwater tank to main steam pipes, plus the reheat line
- Air-flue gas system with flue gas handling and emission control
- Solid fuel feeding from the silos onwards
- Start-up burners from the main oil line onwards
- Ash handling
- Sand feeding
- Chemical injection
- Sampling systems

Besides that, the operator displays revealed that there existed some important equipment not visible on the P&IDs due to being out of the delivery scope, such as sensors for main steam temperature and pressure. The scope of the delivery is important in determining which process areas to monitor extensively, as Valmet is concerned mostly about its own products.

The first thing done was to divide the boiler into logical subprocesses. That way it would be easier to quickly glance which process areas and data sets are applicable to any boiler for which a data collection plan is being compiled. Inside the subprocesses are lists of specified measurements that can be expected to be found within that scope. The list was arranged in the form of a generalized checklist so it wouldn't be tied to any specific boiler configuration: some might have three measurements in some place and others just one, but the list incentivizes to add all measurements applicable to that description. If no applicable measurement or data for a list item exists, it is simply ignored, and likewise it can be expanded upon discovery of a new suitable measurement. As a reference the schematics for two Valmet CYMIC boilers were used, one being the BU4 and the other a separate boiler. As expected, most of the items were applicable to both boilers in some form or another, but the blank spaces showed the unique traits once the data had been compiled. Figure 9 depicts the functional grouping used.

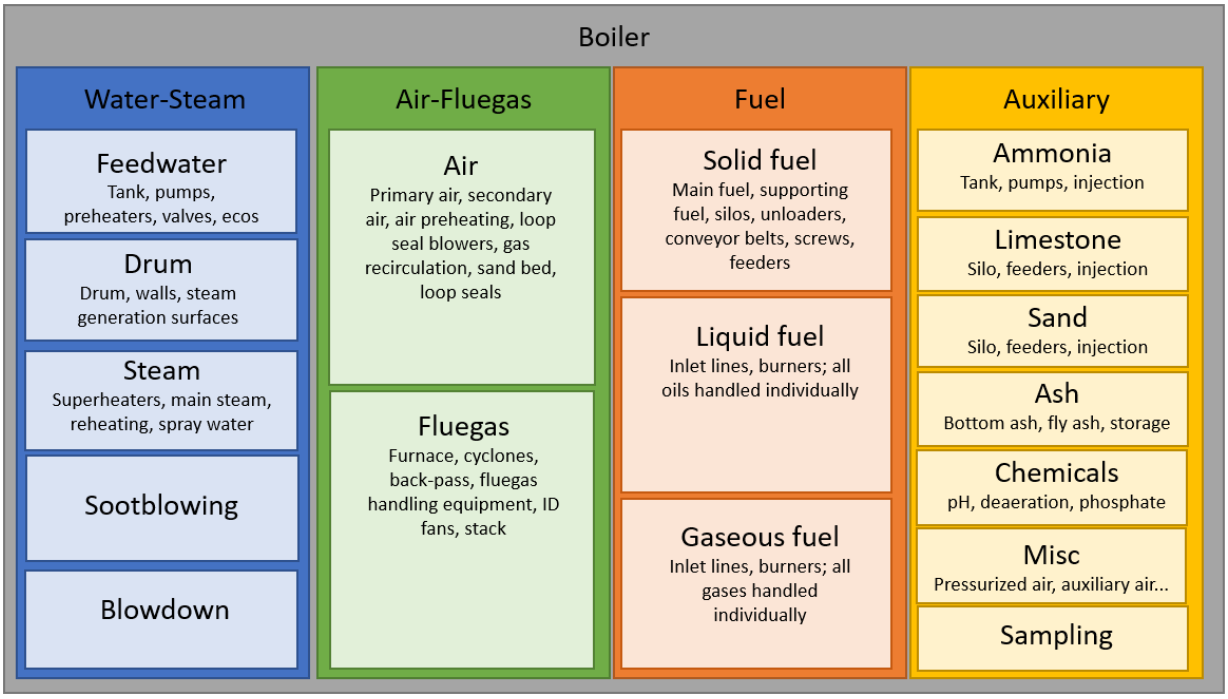


Figure 9. *Subprocesses used in signal grouping, as well as some of the items included.*

When the required measurements were recognized, the final part was to go through the boiler P&IDs and list the necessary sensors. Figure 10 shows a portion of a P&I diagram.

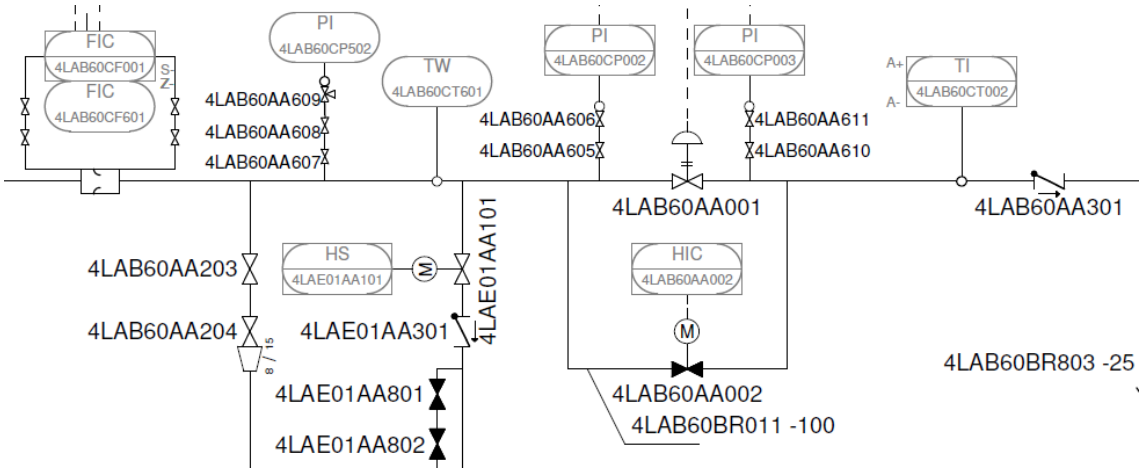


Figure 10. *A portion of a P&I diagram. [74]*

In Figure 10, a section of the feedwater valve system is shown. Each physical component in the field is listed and named according to the KKS. The balloons represent sensors and their types, with PI standing for pressure indicator, TI for temperature indicator, FIC for flow indicator with controller and HIC for valve position indicator with controller. Below the type is the name the sensor is referred to as inside the plant schematics as well as in the DCS. The sensors with a box around them send their measurement values to the DCS and are the sensors that should be looked for, the rest are local indicators such as pressure gauges.

Feedwater	Flow before spray	Flow Controller	4LAB60CF001	pr:4LAB60CF001:av	60
Feedwater	Pressure before FW valve	Pressure 1	4LAB60CP002	pr:4LAB60CP002:av	60
Feedwater	Pressure after FW valve	Pressure 1	4LAB60CP003	pr:4LAB60CP003:av	60
Feedwater	Pressure dp over FW valve	Differential Pressure Controller	4LAB60CP902	pr:4LAB60CP902:av	60
Feedwater	Temperature after FW valve	Temperature 1	4LAB60CT002	pr:4LAB60CT002:av	60

Figure 11. *A portion of the compiled data collection list.*

Figure 11 shows the contents of Figure 10 in a compiled list form. In the list, the first column is the subprocess; the second a measurement identified in the previous step; the third a sensor identifier in case multiple sensors fit the description; the fourth the name of the sensor; the fifth the signal tag name in the DCS carrying the sensor's measurement value; and the sixth the signal collection interval, in seconds. Figure 12 depicts a portion of the list where there is more than one suitable measurement, as well as a measurement with no corresponding sensor.

Sand	Silo level	Sand Silo Level Switch	4EMA10CL101	pr:4EMA10CL101:av	60
Sand	Silo weight	Weight 1	4EMA10CW001	pr:4EMA10CW001:av	60
Sand	Pneumatic transmitter level switc	Level Switch 1	4EMA20CL101	pr:4EMA20CL101:av	60
Sand	Pneumatic transmitter pressure sv	Pressure Switch 1	4EMA20CP102	pr:4EMA20CP102:av	60
Sand		Pressure Switch 2	4EMA20CP103	pr:4EMA20CP103:av	60
Sand		Pressure Switch 3	4EMA20CP104	pr:4EMA20CP104:av	60
Sand	Screw speeds	-	-	-	-

Figure 12. *Another portion of the data list. The screw is omitted due to the boiler having a pneumatic transmitter system instead.*

After the P&IDs had been gone through, next up was the DCS signal part. For this the INT and CON schematics were used. Control schematics describe the relationships between measurements and controllers, show how the signals are handled and relayed between machines and what limits, alarms, modification functions etc. they should have. Control schematics are an abstract depiction of both the process, with only the relevant parts shown, and the automation system, describing the desired outcome and intended behavior. Figure 13 shows a portion of a control schematic.

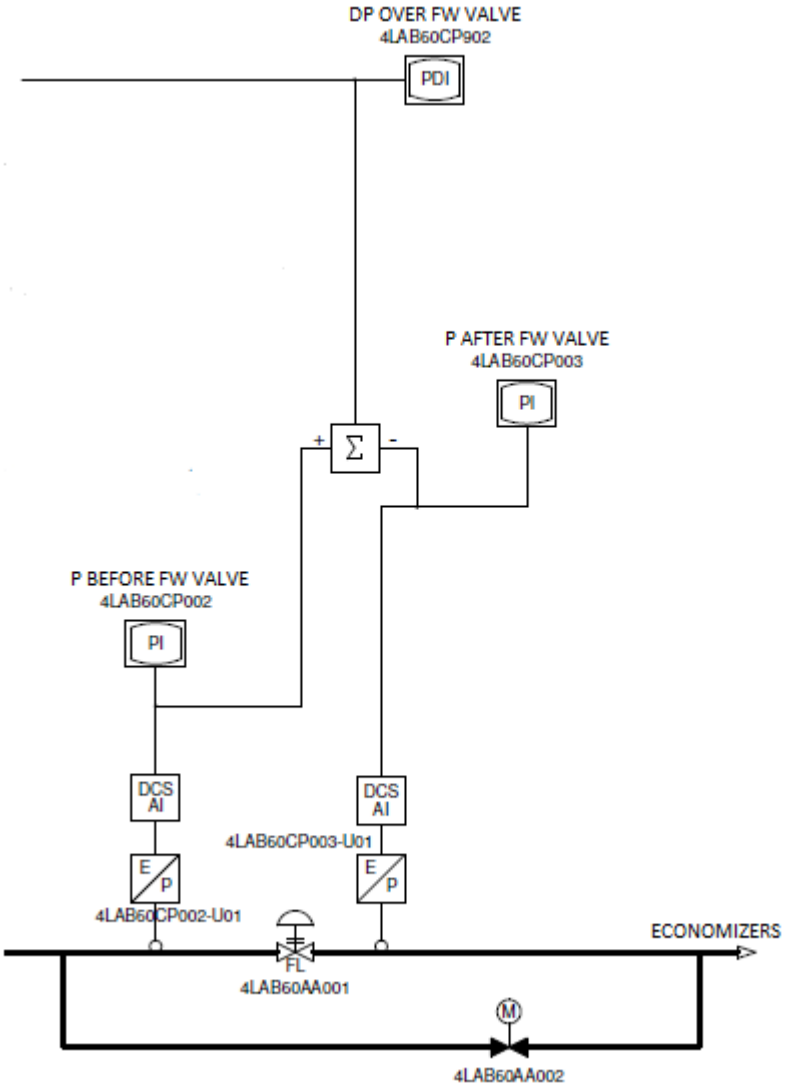


Figure 13. *A portion of a feedwater valve control diagram. The pressure difference (4LAB60CP902) is formed from the pressures before and after the valve. [75]*

INT diagrams describe the regular and safety interlocking logic in the system. Typically, the regular interlocking is carried out in the automation system and handles the normal operating conditions, while safety logic is hardwired to fire off in the event of accidents, alarms and other hazardous situations. Interlockings ensure that neither the operator nor the automation system can make haphazard acts during operation, such as abruptly stop the feedwater pumps during full load. It also ensures the right procedures are taken and can take control of the process if the system detects too large deviations, possibly leading to the shutdown of the process. Figure 14 shows a portion of an interlocking diagram.

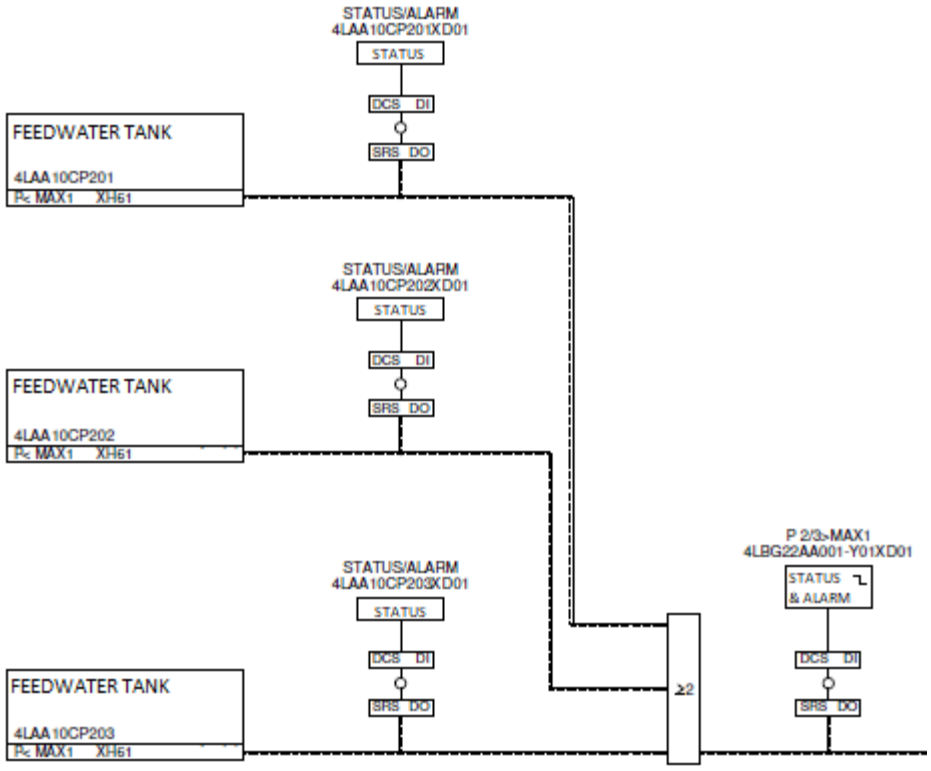


Figure 14. *A portion of an interlocking diagram. The system gives an alarm if 2 out of 3 feedwater tank pressure measurements go over the limit. [76]*

Finally, the cloned automation system revealed how the equipment, controllers and interlockings handle the signals and by what tag name they should be searched. Below are three examples of the different modules in the plant's DCS, running on FbCAD. Figure 15 shows a part of a motor control module, more precisely the control module of feedwater pump 1. The **ins**-port ("in service") contains a binary value and will receive a value of 1 when the automation system detects a signal from the field telling that the motor is running.

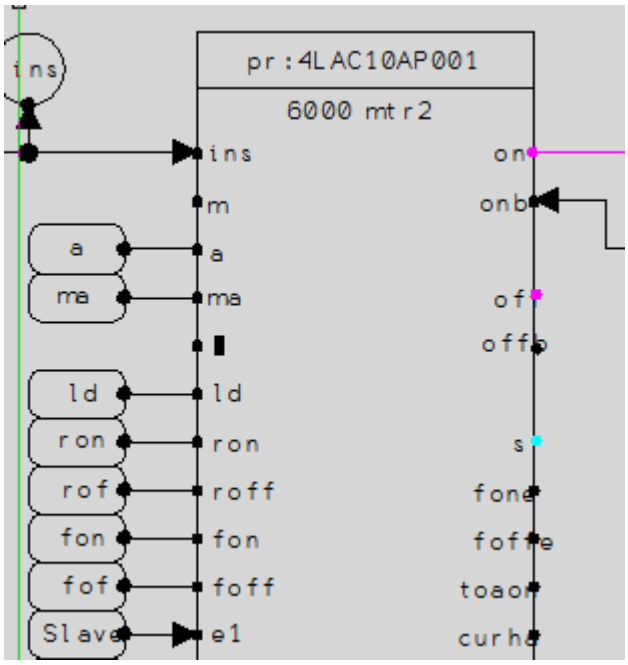


Figure 15. *A section of a motor control module.*

Figure 16 shows a portion of a PID (*Proportional-Integral-Derivative*) controller. The **me**-port contains the measurement value and is the process value the controller is set to control. From the **con**-port the control signal is sent forward to the actuator of the controlling equipment, and the position the actuator reports back arrives in the **pos**-port. Ports **sp1**, **sp2** etc. receive different setpoints that the measurement should achieve, and the **spa**-port tells the currently active setpoint.

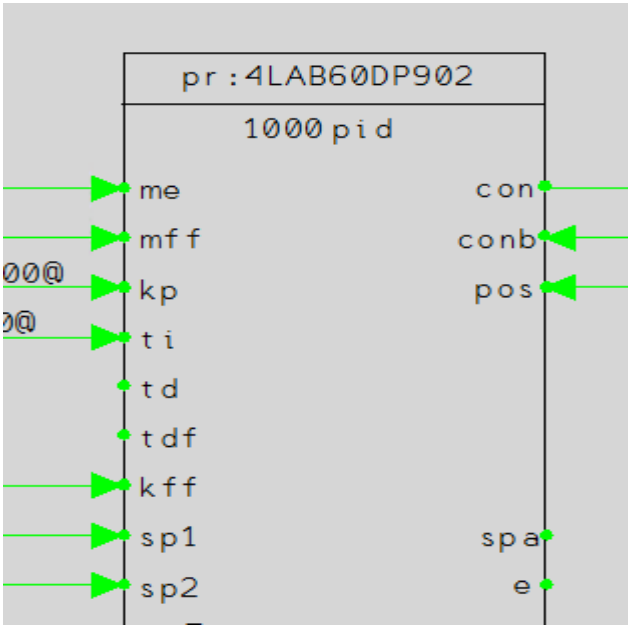


Figure 16. *A section of a PID controller.*

Lastly, Figure 17 contains a direct access port for the boiler main interlock signal. The signal is a timestamped binary value, meaning that whenever its value changes, it also saves the change time. For that reason the input port needs an additional specifier **:binstat** to only store the binary value of the signal.

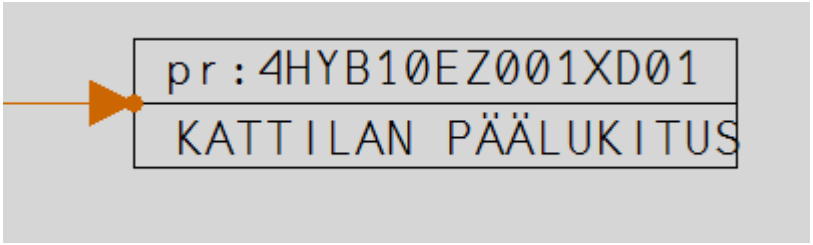


Figure 17. *A direct access port receiving an interlocking signal.*

Figure 18 shows how the above signals are arranged in the data lists. On the DCS signal side there is no grouping for the signals, merely a description for the required signal and the corresponding instance, if applicable.

Motor running status	FW Pump 1 Running	4LAC10AP001	pr:4LAC10AP001:ins	60
Controller measurement	FW Valve Dp	4LAB60DP902	pr:4LAB60DP902:me	10
Controller setpoint	FW Valve Dp	4LAB60DP902	pr:4LAB60DP902:spa	10
Controller position	FW Valve Dp	4LAB60DP902	pr:4LAB60DP902:pos	10
Boiler main interlock		4HYB10EZ001XD01	pr:4HYB10EZ001XD01:binstat	60

Figure 18. *A section of the data collection list depicting different forms of DCS signal naming.*

It should be noted that without access to the DCS it would have been impossible to tell by what name the signals should be referred to. This is not a problem with plants running a Valmet-supplied DCS, but might cause conflicts on plants running a DCS from another supplier.

The final result of this step was a data collection list of some 1 700 tags. Nowhere near all of this data is used in KPI calculations, but all data required by them is definitely there. After all, the data is not used just for KPI calculation, but overall boiler performance assessment, making the raw data just as useful. Of this data about 700 tags were physical measurements, the rest DCS signals. Roughly 30 signals were gathered from outside Valmet scope.

3.2 KPI Definition

As described in chapter 2.7, KPIs are tailor-made to accommodate the needs of the company. Some boiler KPIs are universal, others applicable to a specific type and still others only to a specific configuration of that type. The pilot boiler is a CYMIC-type CFB boiler, so this step concentrates on KPIs mostly applicable to CFB boilers.

Valmet already has internal lists for KPIs for its own products. For example, the CYMIC boiler has some 250 KPIs already listed, ranging from simple measurements to heat transfer performance assessment, mass-energy balance definition and efficiency calculation. For the pilot project a brief selection was gathered with the intention of implementing the rest later when the platform and tools were found adequate. (At this part of the project, the calculation tools were not yet defined.) The following KPIs were selected for the case study of this thesis.

3.2.1 Steam Load Percentage

Steam load as percentage of the MCR. MCR stands for *maximum continuous rating*, the highest rate at which power can be produced continuously without interference. [77] Load percentage tells the overall utilization of the boiler: ideally every customer would like to run their process close to the MCR to generate maximum profit, but the load demand generally is not continuously full. For example, district heating demand is naturally lower during the warm seasons. The KPI is calculated as

$$Load\% = \frac{total\ steam\ load\ [MW]}{MCR\ [MW]} \cdot 100\ \% \tag{12}$$

Depending on the boiler setup, the total steam load may be the sum of main and reheat steam loads.

3.2.2 Stoppage Count

A stoppage is when the boiler unit ceases to produce steam. Stoppages can occur for a number of reasons, such as scheduled maintenance, interlocking trip, accident or a fault at another part of the plant. A high number of stoppages is undesirable for obvious reasons, as the boiler won't produce any value and can take a while to get back online. [78] A stoppage is defined here via a hysteresis: two limits for the load are set, and if the load goes above and then below those limits, a stoppage is recorded. Setting only one limit might cause problems if the process were to be run near the limit for some reason, causing numerous false indications.

3.2.3 Fuel Trip Count

A fuel trip is any event where fuel feeding is interrupted because of interlocking logic. This may happen for example in a case where fuel quality suddenly changes causing a spike in bed temperature, or the drum water level falls too low. Fuel trips cause disruptions in production, and if numerous, the reasons should be traced. [79] The trip count is defined here via five interlocking signals: **bed main interlock** (A), **bed interlock for coal** (B), **solid fuel firing interlock** (C), **any solid fuel inserters running** (D), and **any coal inserters running** (E). If either of the interlocks A or C fail, they automatically bring

the feeders offline, causing D to fail. Additionally, if B fails, coal feeding is shut down, causing E to fail. Any of these events is count as a fuel trip. Also, if a fuel trip is detected, no new trips will be logged for the next 15 minutes. This is to allow time for the process to restart and not count failed ignitions as fuel trips. 15 minutes was defined with the help of process experts to be a sufficiently long time interval for most cases.

3.2.4 Primary Air Ratio

In the context of a CFB boiler, primary air means the air used to fluidize the sand bed. This air also acts as the main combustion air for the solid fuels. Secondary air is divided in burner air and staging air, where burner air is burned at the start-up burners firing oil or gas and staging air is for completing the combustion of the solid fuels in the upper furnace areas. Other sources of air, such as loop seal air, are not considered here, since only primary and secondary air flows are directed in the furnace, where the main combustion happens. Primary air ratio is important in assessing the combustion conditions in the furnace. A high value during nominal operation means staging is not complete, leading to unburned residuals and NO_x emissions in the flue gases. However, the applicability of this KPI is load-dependent: during low loads the value is inherently higher, since primary air used in fluidization needs a minimum flow to achieve the bubbling effect, even if secondary air is not yet used. [37] The KPI is defined as

$$PAR = \frac{total\ primary\ air\ [\frac{kg}{s}]}{total\ primary\ air\ [\frac{kg}{s}] + total\ burner\ air [\frac{kg}{s}] + total\ staging\ air [\frac{kg}{s}]} \tag{13}$$

3.2.5 Bed Temperature

Bed temperature is an important factor in boiler operation. Too high temperatures cause sintering in the bed sand, and too low temperatures may lead to flame quenching. Additionally, the difference between minimum and maximum measurement values shows the maximum variation in temperature in the bed, leading to thermal stress if the value is too high. Each combustion process has a target range for the bed temperature depending on the fuel diet of the boiler. [80]

The bed usually has multiple temperature measurements to give a good overview of the bed temperature profile, with a total of eight sensors in BU4. Bed average temperature is calculated as

$$T_{bed,avg} = \frac{\sum_{i=1}^{k} T_i}{k} \tag{14}$$

where k is the amount of non-faulty temperature sensors [pcs] and T_i the measured temperature of sensor i [°C]. Faulty measurements are automatically ignored by the DCS. The temperature variation is calculated as

$$\Delta T = T_{max} - T_{min} \quad (15)$$

where T_{max} and T_{min} are the highest and lowest non-faulty temperature measurement, respectively [°C].

3.2.6 Own Power Consumption for Combustion Process

Every process requires some electricity to keep the process equipment running. An electricity-generating plant uses a portion of the generated electricity for the upkeep of itself, and a high own power consumption means a less efficient overall process. [81] The plant scope includes every component on the site, but this KPI only concentrates on the boiler, which is the Valmet scope. As such, the KPI is *own power combustion for combustion process*, which is then measured against the produced net energy of the boiler. This gives an indication about how many megawatts of electrical energy the boiler takes to generate one megawatt of heat, which is a much more indicative figure for boiler performance comparison than raw consumed electricity.

For this KPI, only the most relevant consumers were selected. These are

- 2 Feedwater pumps, ~4000 kW each
- 2 Primary air fans, ~1300 kW each
- 2 Secondary air fans, ~600 kW each
- 1 Recirculation gas fan, ~800 kW
- 6 Loop seal blowers, ~200 kW each
- 2 Flue gas fans, ~1500 kW each

Together these machines account for roughly 96 % of all boiler electricity consumption, making the rest of the machinery a marginal issue. All of these are VFD (*variable frequency drive*) controlled motors, meaning they can be driven at partial loads. The integrated frequency converter calculates the consumed wattage in real-time based on the current, voltage and phase difference in the machine circuitry and relays this information to the DCS. The KPI is calculated as

$$P_{own} = \frac{\sum P_i}{total\ steam\ load} \quad (16)$$

where P_i is the instantaneous power consumption of component i [MW_e] and total steam load is the sum of main and reheat steam loads [MW_{th}]. Note that the unit of P_{own} is MW_e / MW_{th}, or megawatts electric power consumed per megawatts thermal power produced.

3.2.7 Sand Consumption

During the operation of the boiler some sand is continuously lost to the bottom ash removal systems. The sand-ash mixture is run through a filtering system, such as a sieve,

separating fine sand and coarse particles from each other, allowing some of this sand to be returned to the furnace. The portion that cannot be returned must be compensated for by injecting fresh sand to the combustion chamber. Sand consumption is tied closely to the fuel load, since an increasing amount of fuel injected also means an increased amount of bottom ash ejected. For that reason, sand consumption should be measured against fuel load. [80]

Sand delivery systems vary between boilers, with BU4 having a pneumatic transmitter system. Below the sand silo is a sealable hopper connected to the pressurized air system. To add sand to the furnace, the top hatch is opened to allow sand to fill the hopper. After the hopper if full, the top hatch is closed and the bottom hatch opened, and pressurized air is used to deliver the contents to the furnace. There is no sensor for measuring the mass flow of these suspended sand particles in the process, and even if there were, the results would show gaps due to the periodic nature of the delivery system. As such, this KPI is day-averaged, as opposed to the others being instantaneous values.

The automation system has a built-in logic for counting the times the transmitter has sent sand to the furnace. The counter resets every night at 0:00, storing the last 7 days' values in an array. Since the pneumatic transmitter only sends the sand forward when the hopper is full, the daily sand consumption then becomes

$$daily\ sand = x\rho_{sand}V_{hop} \tag{17}$$

where x is the transmitter counter at 23:59 that day [pcs], ρ_{sand} is the bulk density of sand [kg/m^3] and V_{hop} the volume of the hopper [m^3].

The DCS automatically calculates the fuel load of the boiler, and in the cloud, it is stored as a minute-averaged value in megawatts. The daily fuel energy consumed is

$$E_{fuel,day} = \frac{1}{3600}\sum_{i=1}^{1440}(\Phi_i \cdot 60) \tag{18}$$

where Φ_i is the minute-averaged fuel load (= power) during minute i [MW], 60 is the number of seconds in a minute (for MW to MJ conversion), 1440 is the number of minutes in a day and 1/3600 is to convert MJ to MWh. The KPI is then

$$sand\ consumption = \frac{daily\ sand\ [kg]}{E_{fuel,day}} \tag{19}$$

where $E_{fuel,day}$ is in megawatt-hours [MWh].

3.2.8 Emissions

Emissions are another strictly-monitored parameter in a boiler, most notably due to local regulations. In Finland authorities require periodic reports about industrial emissions, and

the reports should conform to strict standards. Regulated species include carbon monoxide (CO), oxides of nitrogen (NO_x) and sulfur dioxide (SO_2). [82]

Besides harmful substances, the analyzed species also include flue gas end oxygen (O_2) content as well as the flue gas moisture (H_2O). O_2 is the leftover oxygen from the combustion process due to not using a stoichiometric air/fuel ratio, but rather a bit higher one. The oxygen content in flue gas usually has some target value, because too low could lead to incomplete combustion and too high is an indicator of using too much excess air, leading to a less efficient process. (The air needs to be pumped, heated and exhausted, each action requiring energy.) The H_2O is the sum of three sources: humidity in the combustion air, moisture in the fuel and H_2O formed during the combustion process. Moisture in the flue gas in unavoidable but undesirable, because a portion of the chemical energy released during combustion is bound in the water as *latent heat*, the heat it takes to evaporate the water. Flue gases are usually exhausted at temperatures well over 100 °C to avoid the acid dew point temperature, where acidic compounds in the flue gas start to condense on the surfaces of the flue gas ducts. Thus, the latent heat is lost from the process along with the moisture. [24, 83]

The flue gas analyzers are located at the stack to give an accurate view about the compounds ending up in the atmosphere. Here all the measurements are "wet" measurements, meaning they are analyzed from the complete flue gas flow including the moisture. A wet concentration measurement can be converted to dry basis as follows [24]:

$$c_{x,dry} = c_{x,wet} \cdot \frac{100}{100 - c_{H_2O}} \tag{20}$$

where $c_{x,wet}$ is the measured concentration of species x [mg/m^3] and c_{H2O} is the moisture measurement [%].

All measurements must also be reduced to a reference O_2-value specified for each boiler. During nominal load and firing design fuel the BU4 has a reference O_2-value of 6 %, meaning all concentrations need to be reduced as follows [24]:

$$c_{x,dry,ref} = c_{x,dry} \cdot \frac{21 - c_{O_{2,ref}}}{21 - c_{O_{2,dry}}} = c_{x,dry} \cdot \frac{15}{21 - c_{O_{2,dry}}} \tag{21}$$

where $c_{x,dry}$ is the dry-basis concentration of species x [mg/m^3], $c_{O2,ref}$ is the reference O_2 for that plant [6 %] and $c_{O2,dry}$ the measured dry oxygen content at stack [%].

The measurement still needs to be converted to standard conditions, since gas density and volume are highly influenced by its temperature and pressure. In order to compare values from different plants with different stack conditions, a standard conversion is used. Standard state is defined as 273 K and 101350 Pa. The conversion is [24]:

$$c_{x,dry,ref,N} = c_{x,dry,ref} \cdot \frac{(273+T)}{T_{ref}} \cdot \frac{p_{ref}}{(1.0135+p)} = c_{x,dry,ref} \frac{(273+T)}{273} \cdot \frac{1.0135}{(1.0135+p)} \quad (22)$$

where $c_{x,dry,ref,N}$ is the dry-basis, O_2-referenced and reduced concentration of species x [mg/Nm^3], $c_{x,dry,ref}$ is the dry-basis, O_2-referenced concentration of species x [mg/m^3], T_{ref} is the standard temperature [273 K], p_{ref} the standard pressure [1.0135 bar], T the measured stack temperature [°C] and p the measured stack pressure [bar] as gauge pressure, hence the addition of one atmospheric pressure (1.0135 bar). [24]

3.2.9 Excess Air Coefficient

As stated previously, the combustion process requires more than the theoretical amount of air present to account for the oxygen that goes unreacted. This is represented with the *excess air coefficient* λ, a factor larger than 1 indicating the amount of air used opposed to the theoretical requirement. For simple fuels, such as methane (CH_4), it is possible to devise analytic relations for theoretic air requirements via combustion reactions, and calculate the excess coefficient from measured air flow. With more complex fuels such an approach is not practical, as the true fuel composition is never really known. [24]

However, there is a rather simple workaround applicable to both simple and complex fuels. Atmospheric air contains roughly 21 % of O_2 by volume. With this in mind, if the O_2 content of the flue gases after the furnace is measured, the following must hold true [24]:

- O_2 content = 0 % → stoichiometric or air deficient combustion
- O_2 content = 21 % → no combustion
- O_2 content between 0 % and 21 % → combustion with excess air

Increasing O_2 content also means an increasing excess air coefficient, which can be calculated as

$$\lambda = \frac{21}{21 - c_{O_{2,dry}}} \quad (23)$$

where $c_{O2,dry}$ is the measured dry-basis oxygen content of the flue gases in the back-pass [%]. The dry conversion is done from the wet measurement with equation (11) using the stack moisture. The same logic explains the number 21 in equation (21). [24]

3.2.10 Water Chemistry

The main working fluid of a boiler is water, present in both liquid and steam forms. The water experiences high temperatures, pressures and flow rates throughout the process, interacting with different components as it flows. Piping is the most susceptible part of the infrastructure to experience stress from the flowing water, making water chemistry an

important consideration to prevent leaks and clogging. Water chemistry is monitored with a sampling system, where outtakes are lead from relevant process points. [84] Three water chemistry related values are included in the KPI set to showcase different process areas and quality criteria.

O_2 in feedwater analyzes the oxygen gas dissolved in the feedwater leaving the tank. O_2 causes corrosion and rusting with iron-based piping, creating either ferric hydroxide ($Fe(OH)_3$ and $FeO(OH)$) at low temperatures and ferric oxide (Fe_2O_3) at high temperatures. O_2 removal (called *deaeration*) is conducted in the feedwater tank as thermal deaeration, where a high-temperature extraction flow is directed from the turbine to the bottom of the feedwater tank. When the feedwater comes in contact with the steam it is quickly brought to saturation temperature, causing dissolved gaseous species like CO_2 and O_2 to be released and subsequently vented. O_2 content is measured as a concentration, either as mg O_2 / kg water or ppb (parts per billion). [84]

Boiler water pH analyzes the acidity of the water in the drum-wall circulation. pH control is crucial to the health of the process, as too low pH causes strong corrosion on the steel pipes and too high pH leads to foam formation. Processes have a "safe zone" of acidity depending on the process nature, and that range should not be exceeded. pH is controlled with injectable chemicals, such as sodium hydroxide (NaOH), a strong base. For the pH measurements to be comparable the samples need to be cooled to a reference point, 25 °C, where pure water has a pH of roughly 7. The cooling system is a part of the sampling system. [84]

Main steam conductivity analyzes the electric conductivity of the steam. Electric conductance is a result of different ion-form impurities, known as *salts*, dissolved in the water. It is measured in millisiemens/meter, mS/m, and is a cheap and simple tool for analyzing the quality of water. Conductivity measurements can be either direct or indirect. In direct measuring the sample is analyzed as it is. In indirect measuring different volatile chemicals injected during the process are removed, leaving only the salt-laden water. Abnormally high conductivities can be a result of a leakage or defective water handling systems. High conductivity can be amended with either increased blowdown rates or replacing a portion of the boiler water with new water. [84]

3.2.11 Recirculation Gas Fan Usage

Flue gas recirculation (FGR) is a process where a portion of the stack flue gases is directed back to the furnace. The flue gases are mixed in with combustion air (in this case the staging air), lowering the O_2 concentration in the oxidizer stream. This causes regions of low O_2 to form in the furnace, limiting the maximum flame temperature. Both causes lead to decreased NO_x formation, as there is not enough O_2 for the nitrogen and the temperature is not that high. The obvious downside of this is increased CO formation, resulting from O_2-deficient combustion, as well as increased instability of the combustion process

and loss of firing capacity. [85] The FGR fan is the third-largest fan after the flue gas and primary air fans and is completely optional, so the benefits of this costly system with such downsides should be measured carefully. Here the KPI is to indicate FGR fan usage over a long period to track whether or not the investment has been worthwhile. It tracks both the start-up count as well as the total running time when the boiler has been firing.

3.2.12 Mechanical Vibrations

Rotating pumps and fans have blades fitted along the rotational axis which interact with the pumped fluid. These blades can be straight or curved, and are shaped to cause the intended velocity and pressure increase in the fluid as it travels through the blade sections. With VFD controlled motors changing load directly affects the speed at which the blades rotate, with a relayed effect on the fluid.

For every structure exists a frequency, known as *natural frequency*, at which the structure begins to resonate. Resonance causes high-amplitude vibrations which can lead to mechanical fatigue, stress and ultimately failure of integrity. For rotating equipment, the frequency comes from the rotation speed of the shaft. Operating the machinery near the natural frequency or its harmonic multiples can cause several factors higher wear and tear than during normal operation, reducing the lifetime of the equipment. Vibration can also be caused by shaft misalignment resulting from bending or improperly seated bearings, or passing the critical speed of the machine configuration. Mechanical vibration is measured at two points of the pump, the motor-side bearing and pump-side bearing, called DE (Drive End) and NDE (Non-Drive End) sides respectively. It is measured in millimeters/second (mm/s) indicating the velocity at which the shaft is moving inside the seating. [86]

Tracking the mechanical wear is an obvious area of interest with respect to maintenance requirements and operational smoothness. The KPI calculates the amount of time the NDE measurement has been over the specified safe range for feedwater pumps, primary and secondary air fans, flue gas recirculation fan and induced draft fans, which can be used to both estimate equipment maintenance needs and spot possible failures.

3.3 Data Pipeline Implementation

The data gathering procedure can be broken down in four steps:

- Opening the data pipeline
- Data gathering from source systems
- Metadata configuration
- Data transfer and storage

In the case of Valmet-supplied DCS systems, the source system is a historian database located on a separate computer known as the INFO computer. Here the continuous process values generated by the DCS are stored in a good fidelity for a specified time. The primary role of the INFO computer has thus far been to provide insight to plant conditions for Valmet experts, to help with support requests. The VII will in its part take this function to the next level.

To open the pipeline, the source and destination systems first need to be opened to each other. AWS natively rejects all non-authorized incoming data, and the INFO can only send data to select destinations. The source system configuration is left for customer IT, while AWS is handled by Valmet. The customer is set up with an SFTP account in AWS where all the data from their plants arrive. From the source system it would be possible to peek into the contents of the destination system, so each customer is assigned their own account, the contents of which only they'll see. After this the connection is tested and the credentials are saved on the source system.

Data gathering is done via dedicated scripts running on the INFO machine. The scripting language used is R, which first needs to be installed and configured on the INFO. After this is done, the historian database needs to be opened to allow the scripts access. This is done via setting up an ODBC (*Open Database Connection*) exception in the database for the scripts.

After the database is open, a series of premade scripts is employed for data extraction. The scripts come in two flavors: one for continuous data dumping and one for gathering history data. The history script is configured to gather data once from a specified date to the current day, while the continuous script is given a time interval for continuous dumps. In both scripts, the following data is entered:

- millid: Valmet internal designation number for the source site
- lineid: Valmet internal designation number for the source line (i.e. different boiler units)
- business: business segment of the customer company
- source: name of the INFO machine in the local system
- datatype: a specifier for the data, tells where the data should be located in the AWS system

The requested data for collection is located in a different file, which is a plain text file containing the names of the signals compiled in chapter 3.1.

When the script is configured and tag lists are added to the text file, the script can be run. The result will be two files in the following format:

```
millid_segment_lineid_source_datatype_timestamp.csv
millid_segment_lineid_source_datatype_timestamp.ini
```

The configured time interval controls the timestamp creation. If set for 10-minute intervals, the minute-averaged values of the last 10 minutes are stored in the same file, and the timestamp is then set as the next ten minute mark. The actual values are contained in the .csv (*Comma Separated Values*) file in the following format:

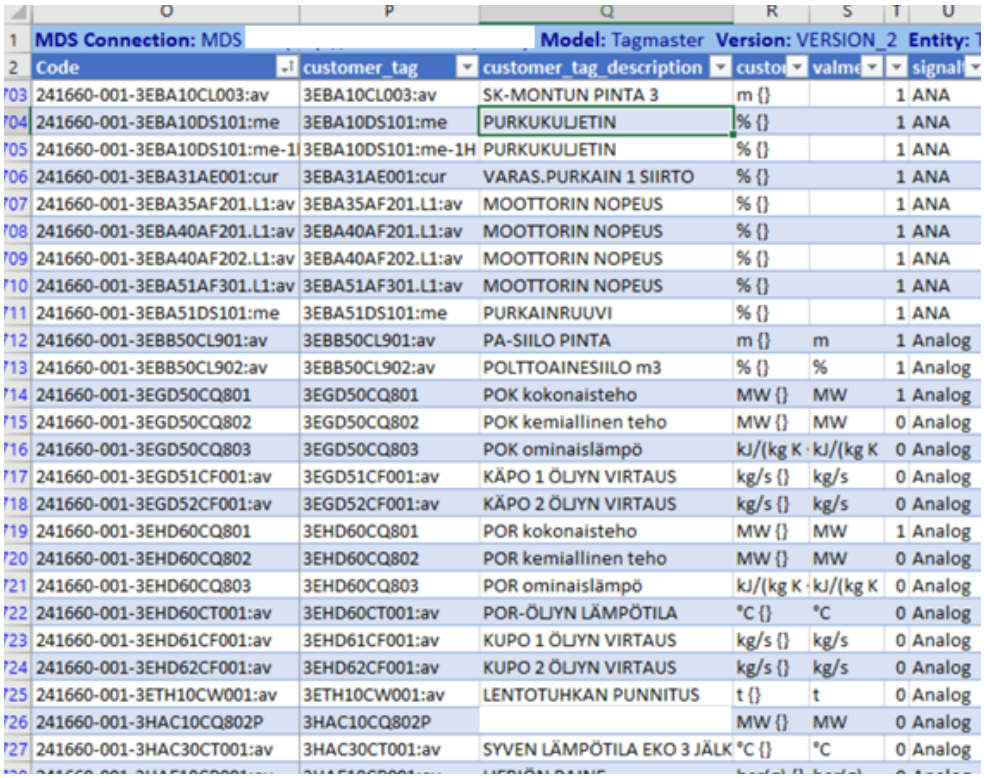
```
scan_dttm,tag_name,value
2017-10-14 9:00:00,pr:4LBA10CJ901:av,337.066
2017-10-14 9:01:00,pr:4LBA10CJ901:av,336.447
2017-10-14 9:02:00,pr:4LBA10CJ901:av,335.324
…
```

Each CSV file contains the minute averages of the day for each tag specified in the data collection plan. The .ini file is a control file which records the start and end time of collection as well as total number of rows written.

Before the data can be sent, the destination system needs to be told what sort of data will be arriving. For this the MDM will be utilized. First, the source system historian is probed for metadata about the signals set for collection. This information includes

- Description, which is the same as the one appearing in the DCS for the signal
- Unit used for the signal in the local DCS, e.g. MW for generator load
- Signal type: analog or discrete

After this information is collected, it can be recorded in the MDS (*Master Data Service*, a module of MDM). This can be done via an Excel plugin called Tagmaster, shown in Figure 19.



	O	P	Q	R	S	T	U
1	MDS Connection: MDS			Model: Tagmaster	Version: VERSION_2		Entity: T
2	Code	customer_tag	customer_tag_description	custor	valme		signal
703	241660-001-3EBA10CL003:av	3EBA10CL003:av	SK-MONTUN PINTA 3	m {}		1	ANA
704	241660-001-3EBA10DS101:me	3EBA10DS101:me	PURKUKULJETIN	% {}		1	ANA
705	241660-001-3EBA10DS101:me-1I	3EBA10DS101:me-1H	PURKUKULJETIN	% {}		1	ANA
706	241660-001-3EBA31AE001:cur	3EBA31AE001:cur	VARAS.PURKAIN 1 SIIRTO	% {}		1	ANA
707	241660-001-3EBA35AF201.L1:av	3EBA35AF201.L1:av	MOOTTORIN NOPEUS	% {}		1	ANA
708	241660-001-3EBA40AF201.L1:av	3EBA40AF201.L1:av	MOOTTORIN NOPEUS	% {}		1	ANA
709	241660-001-3EBA40AF202.L1:av	3EBA40AF202.L1:av	MOOTTORIN NOPEUS	% {}		1	ANA
710	241660-001-3EBA51AF301.L1:av	3EBA51AF301.L1:av	MOOTTORIN NOPEUS	% {}		1	ANA
711	241660-001-3EBA51DS101:me	3EBA51DS101:me	PURKAINRUUVI	% {}		1	ANA
712	241660-001-3EBB50CL901:av	3EBB50CL901:av	PA-SIILO PINTA	m {}	m	1	Analog
713	241660-001-3EBB50CL902:av	3EBB50CL902:av	POLTTOAINESIILO m3	% {}	%	1	Analog
714	241660-001-3EGD50CQ801	3EGD50CQ801	POK kokonaisteho	MW {}	MW	1	Analog
715	241660-001-3EGD50CQ802	3EGD50CQ802	POK kemiallinen teho	MW {}	MW	0	Analog
716	241660-001-3EGD50CQ803	3EGD50CQ803	POK ominaislämpö	kJ/(kg K	kJ/(kg K	0	Analog
717	241660-001-3EGD51CF001:av	3EGD51CF001:av	KÄPO 1 ÖLJYN VIRTAUS	kg/s {}	kg/s	0	Analog
718	241660-001-3EGD52CF001:av	3EGD52CF001:av	KÄPO 2 ÖLJYN VIRTAUS	kg/s {}	kg/s	0	Analog
719	241660-001-3EHD60CQ801	3EHD60CQ801	POR kokonaisteho	MW {}	MW	1	Analog
720	241660-001-3EHD60CQ802	3EHD60CQ802	POR kemiallinen teho	MW {}	MW	0	Analog
721	241660-001-3EHD60CQ803	3EHD60CQ803	POR ominaislämpö	kJ/(kg K	kJ/(kg K	0	Analog
722	241660-001-3EHD60CT001:av	3EHD60CT001:av	POR-ÖLJYN LÄMPÖTILA	°C {}	°C	0	Analog
723	241660-001-3EHD61CF001:av	3EHD61CF001:av	KUPO 1 ÖLJYN VIRTAUS	kg/s {}	kg/s	0	Analog
724	241660-001-3EHD62CF001:av	3EHD62CF001:av	KUPO 2 ÖLJYN VIRTAUS	kg/s {}	kg/s	0	Analog
725	241660-001-3ETH10CW001:av	3ETH10CW001:av	LENTOTUHKAN PUNNITUS	t {}	t	0	Analog
726	241660-001-3HAC10CQ802P	3HAC10CQ802P		MW {}	MW	0	Analog
727	241660-001-3HAC30CT001:av	3HAC30CT001:av	SYVEN LÄMPÖTILA EKO 3 JÄLK	°C {}	°C	0	Analog

Figure 19. *A portion of the Tagmaster Excel plugin.*

Tagmaster contains the metadata collected from the site, as well as some additional information:

- The source site's millid and source line's lineid
- A code made from the millid, lineid and tag name, used as a unique name for the signal in the cloud
- Valmet unit, in case the source site uses non-SI units
- Valmet tag, the generic signal name
- A switch for controlling if the signal should be also sent to Redshift

After all the information has been recorded, the tables are published for updating the MDS. This step needs to be done before any data can be transferred, since the cloud rejects any signals not correctly specified in the MDS.

Once the system is ready, data transfer can begin. The INFO computer is set up with WinSCP, an SFTP client. WinSCP opens a secure tunnel for data transfer between the source and destination systems. The tunnel is set between the location where the scripts will dump the data files and the customer's own S3 bucket, a part of their AWS account. Figure 20 shows the interface of the WinSCP.

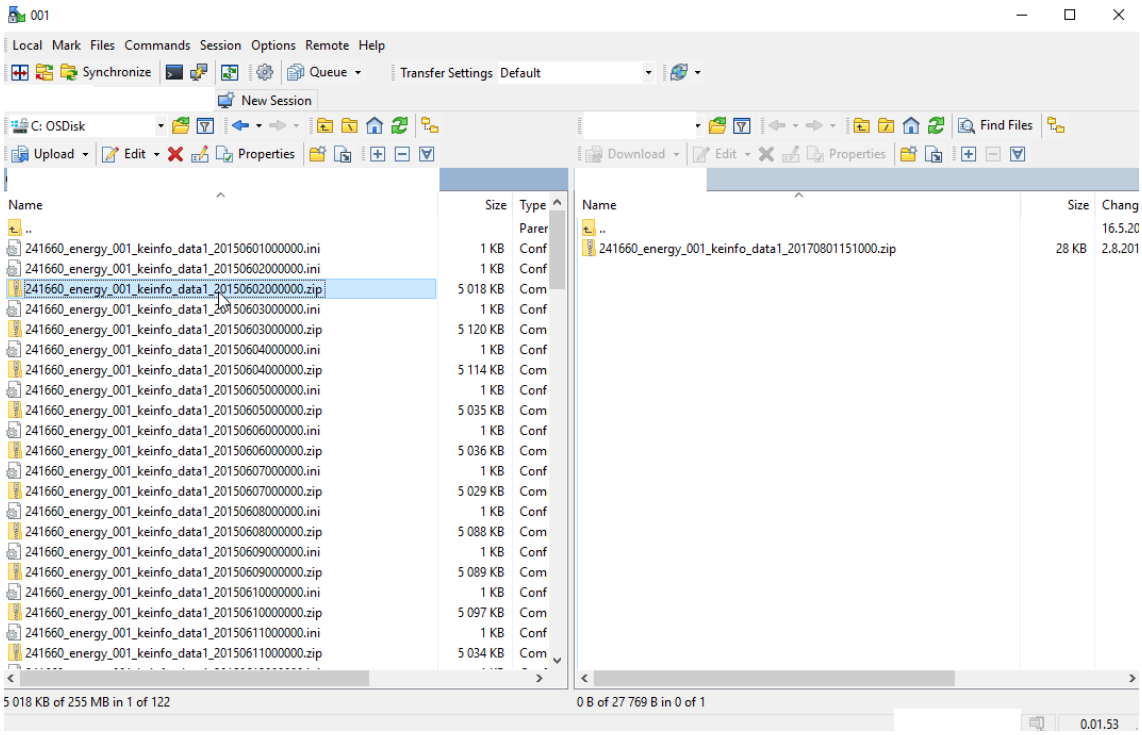


Figure 20. *WinSCP interface, with the source system files on the left and the destination bucket on the right. Sensitive information has been censored.*

An R-script is configured to send the newest files at a specified interval, which can be anything from five minutes to a day. First the CSV files are converted to .zip files to reduce the file sizes. The script then first transfers the .zip files to the designated customer bucket in S3, and the .ini files second. At this point the AWS data handling takes over. Data enters the staging area and goes through the modeling procedure. The .zip files are extracted and the data in the .csv files is converted to .json (*JavaScript Object Notation*) format for faster processing. The data is then integrated in the relevant tables.

On the cloud side there are a few tools for interacting with the data. The first is AWS Console, which is a browser-based file explorer for monitoring and configuring the S3. Here the .zip files can be accessed and downloaded if desired. The files on the S3 are not the original ones, as they have been processed to include additional metadata and to comply with the cloud naming standards. However, the original raw data is still retrievable from them. The files are inserted in day-specified buckets inside the folder designated for each customer, as shown in Figure 21. AWS Console also allows the monitoring of any traffic and scripts running on the system through the script executor queue.

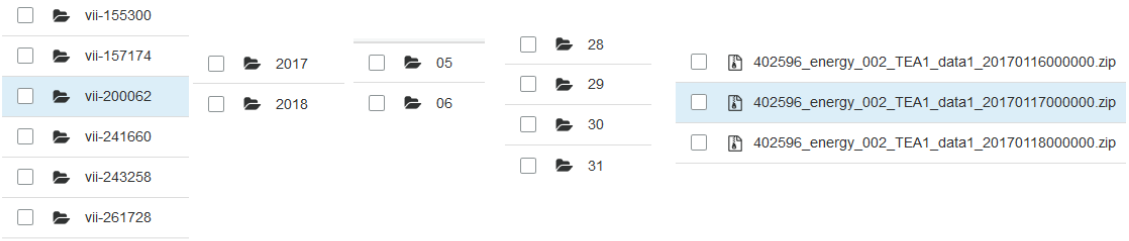


Figure 21. *Logical structure of the AWS buckets. From left to right: customer folders, years, months, days and files for that day.*

Another tool for data manipulation is Aginity Workbench, which is a database tool with a graphical user interface. Here the data can be organized in tables via SQL queries. All the data from Tagmaster and site can be viewed here. The tables in the database are visible in the object browser, the queries can be entered in a query window and the resulting tables can be viewed underneath. Figure 22 shows the layout of the Aginity Workbench.

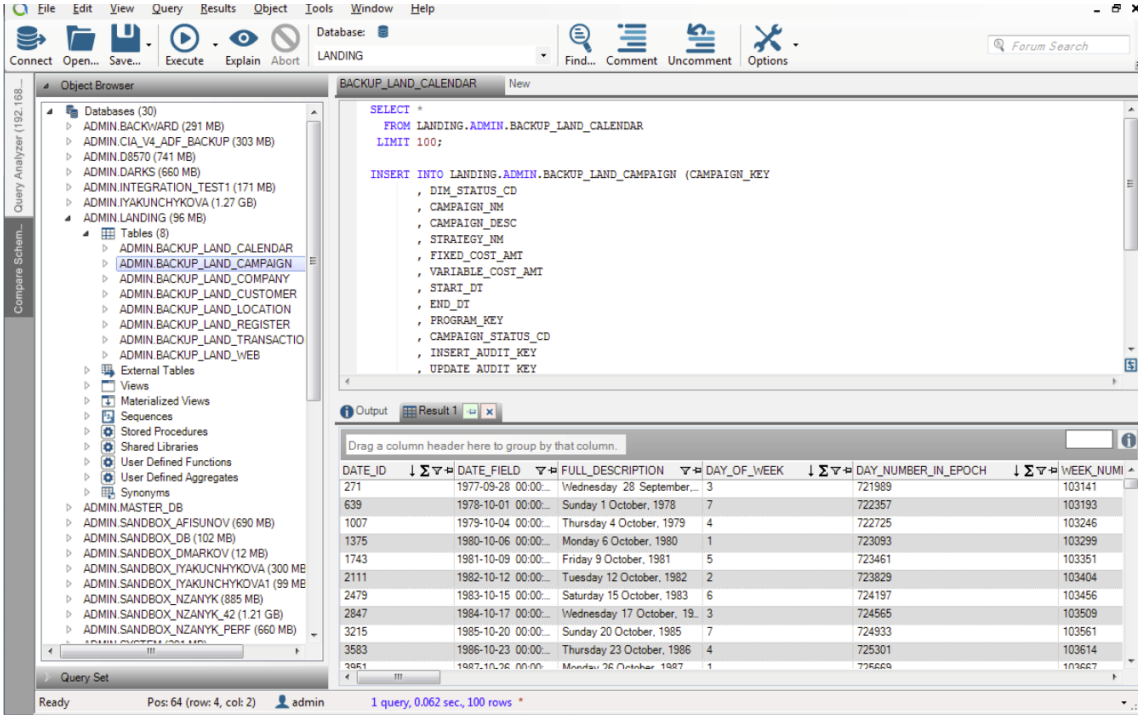


Figure 22. *The layout of Aginity Workbench. With correctly formulated queries, the process data can be viewed on the output panel. [87]*

Aginity has variable versions depending on which environments it's set to run in, with one module being for Redshift. As such, Aginity can only access and view process data which was set to go to Redshift in the Tagmaster.

The third way to work with the data is Jenkins, a web-based automation server. Jenkins allows implementation and scheduling of scripts based on a number of triggering events. The scripts for manipulating the data are located here, and can include scripts for KPI

calculation, aggregating and data vault housekeeping, among others. For example, when the data enters Redshift, it is first pushed to a single table. Data arrival triggers a script in Jenkins that automatically calculates hourly and daily averages, pushes the different values in the right tables and empties the staging table.

3.4 Platform Specifics

In this context, when data is said to be "in the cloud", it means it has been processed through the S3 buckets and modeled in Redshift Data Vaults. Only a portion of the data is sent to Redshift, which can be controlled through the Tagmaster. Redshift data is 'live', meaning it can be accessed and interacted with by both Aginity and Birst, but the data in S3 buckets remains in CSV format. Currently the only way to interact with the data is to download individual files and open them with suitable applications, such as Excel or Notepad.

Data lifecycle refers to the amount of time the data resides in a particular container. As data becomes older its relevancy diminishes, because from an operational point of view only the fresh data is of interest. On the other hand, big data analytics is more fruitful if there is data available from a long range of time. Since VII aims to satisfy both needs, a cascade model of data lifecycle has been developed as a compromise. AWS offers along with S3 and Redshift a third option for data storage, the AWS Glacier, intended as cheap deep storage for massive data sets [88]. Storage space is inexpensive, but the data is not immediately available even if queried. Data lifecycle plan is defined as follows:

- Fresh operational process data goes to Redshift and stays by default for 3 months.
- After 3 months, the data is purged from Redshift. Since all incoming data is additionally recorded in S3, there exists a copy of the data for the first 3 months, after which it only exists in CSV format. Calculation results done with Redshift data (such as KPI values) is saved in S3 after 3 months as well. Data will remain in S3 for 2 years by default.
- After 2 years, the data is transferred to Glacier, where it will remain for an indefinite time. Current proposal for default storage time is 20 years.

With this plan, the interesting information, which most probably is also queried most often, is accessible fast and easily. Then the data is pushed to the intermediate storage along with the less-interesting information, available if the customer would like to see the boiler performance over a year, for example. Finally, the data is pushed to Glacier, where it can be retrieved if required for a large-scale calculation, but it will not burden the faster platforms, since the faster storage platforms also cost more per stored gigabyte. The figures presented are proposed default values and can be negotiated with the customer.

Data ownership is another issue concerning customers. Legally, any data produced by systems under customer jurisdiction (measurements, DCS calculations…) is owned by

the customer. If the customer so desires, Valmet is bound to delete select or even all their data from the cloud. The calculation results, however, are something of a grey area. Derived calculations based on the raw data are used at Valmet for product quality management, and the fate of that data can be negotiated with the customer separately.

Currently, the cloud data can be accessed and viewed in two ways: via SQL queries in Aginity, or by creating dashboards in Birst. From a process engineer point-of-view however, Aginity is not very flexible due to requiring knowledge of SQL and lacking any visualization features. For this reason, Birst will be used until a more suitable solution is found.

To allow data visualization in Birst, a link between the web-based Birst tool and the Redshift database first needs to be established. This is done by creating a Birst "space" for the connection. The space is an isolated area with tailor-made content, such as trends, KPI dashboards and filtering options, and each space only sees data it has been connected to. During the connection phase, the source databases are identified and user groups for the space defined. User groups can include developers, who can alter the content and layout of the space, and viewers, who can only read and download data from the dashboards. Figure 23 shows the relationships between the connected tables.

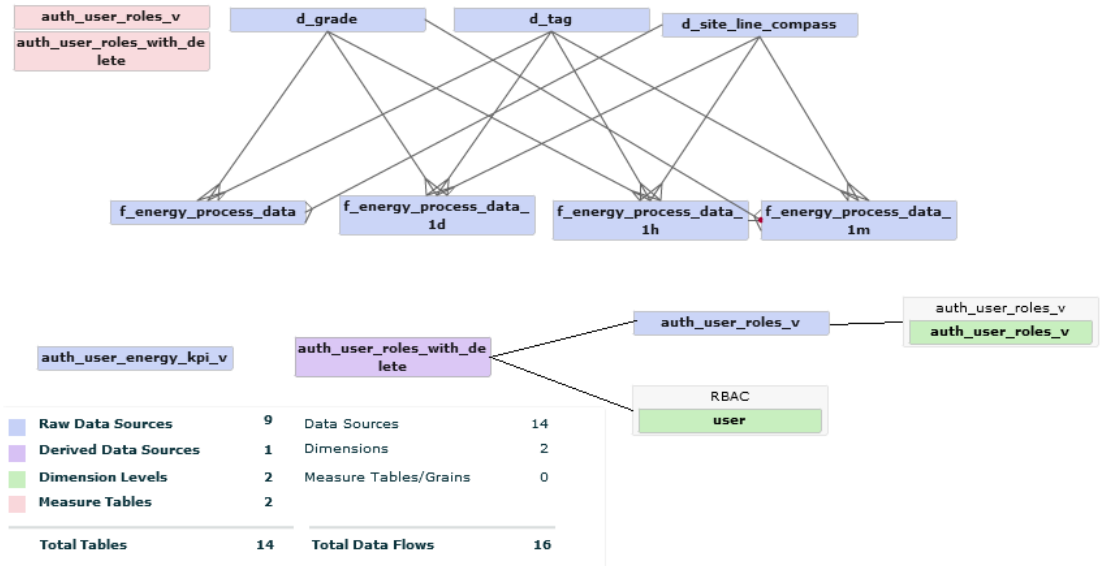


Figure 23. *Defining Birst sources. Some of the lines needed to be redrawn by hand for compact presentation.*

In Figure 23, the arrows define relationships between tables. The top three tables d_grade, d_tag and d_site_line_compass contain metadata, such as units or customer information. The four f_energy_process_data tables contain the actual timestamped signals and their values. The table with the actual sent values as minute averages reside in the _1m table. From these minute averages hour- and day-averaged values are then calculated and readily stored in Redshift as soon as new data arrives. Aggregation could be done in Birst

during visualization as well, but performance is better if the data is pre-calculated. The remaining tables contain user access and interaction rights.

After the Birst space is set up, the developers can begin adding content. This is done in the Visualizer tool, seen in Figure 24.

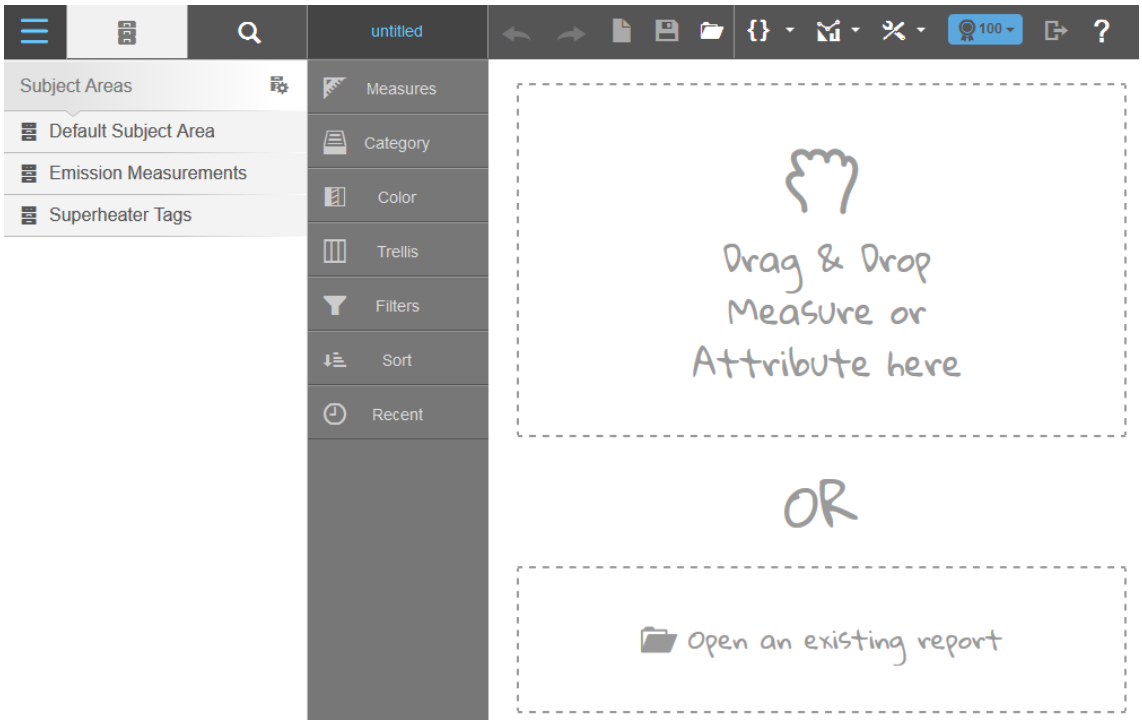


Figure 24. *The Birst Visualizer tool, ready to start making trends.*

To bring data from the database to visualize, the developer can either navigate the left-hand folders or write a query in Birst Query Language (BQL), accessible from the wave brackets icon. BQL is more flexible in what it can do, so it will be used for demonstration. Inside the query window a simple script is written to extract daily averages of process data measurements, where the customer tag matches the description.

```
[timestamp_date: Avg: tag_value_data_1m_] WHERE [d_tag.customer_tag] = '1LBA00DU903:av'
```

Figure 25. *A simple BQL query.*

The data is then measured against the date of the timestamp, and a trend of the values will be presented.

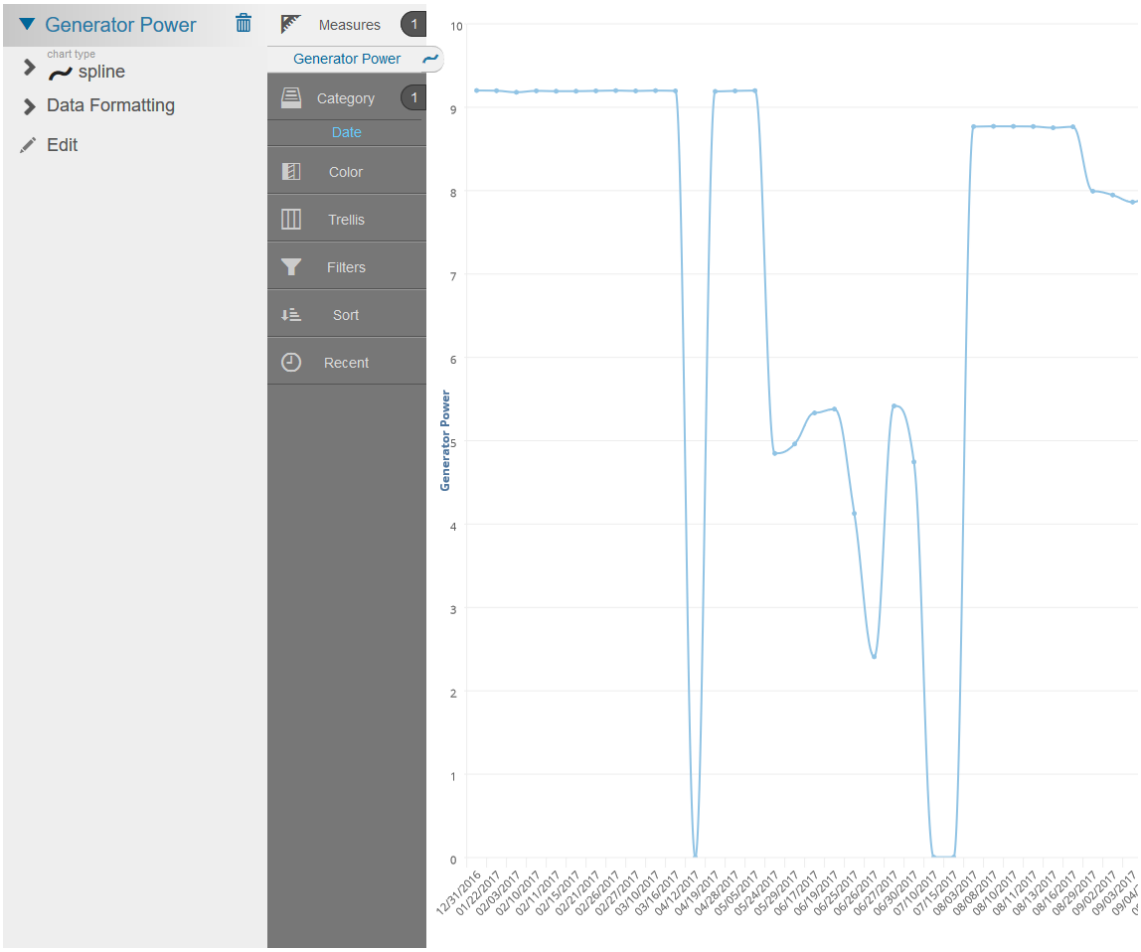


Figure 26. *Visualization of a trend in Birst.*

The trends can be given different forms, such as line, spline or bar charts. It is possible to present multiple trends in the same graph, and even conduct simple arithmetic. In Figure 27, a code for summing up two signals is presented.

```
([timestamp_date: Avg: tag_value_data_1m_] WHERE [d_tag.customer_tag] = '1NDA00DE901:av') +
([timestamp_date: Avg: tag_value_data_1m_] WHERE [d_tag.customer_tag] = '1MKA10CE069:av')
```

Figure 27. *A BQL script for summing two signals in visualization.*

Birst calculates the sum of the signals and presents the output on the visualization layer correctly. Figure 28 shows the electricity generation, district heating load and the sum of both in one trend.

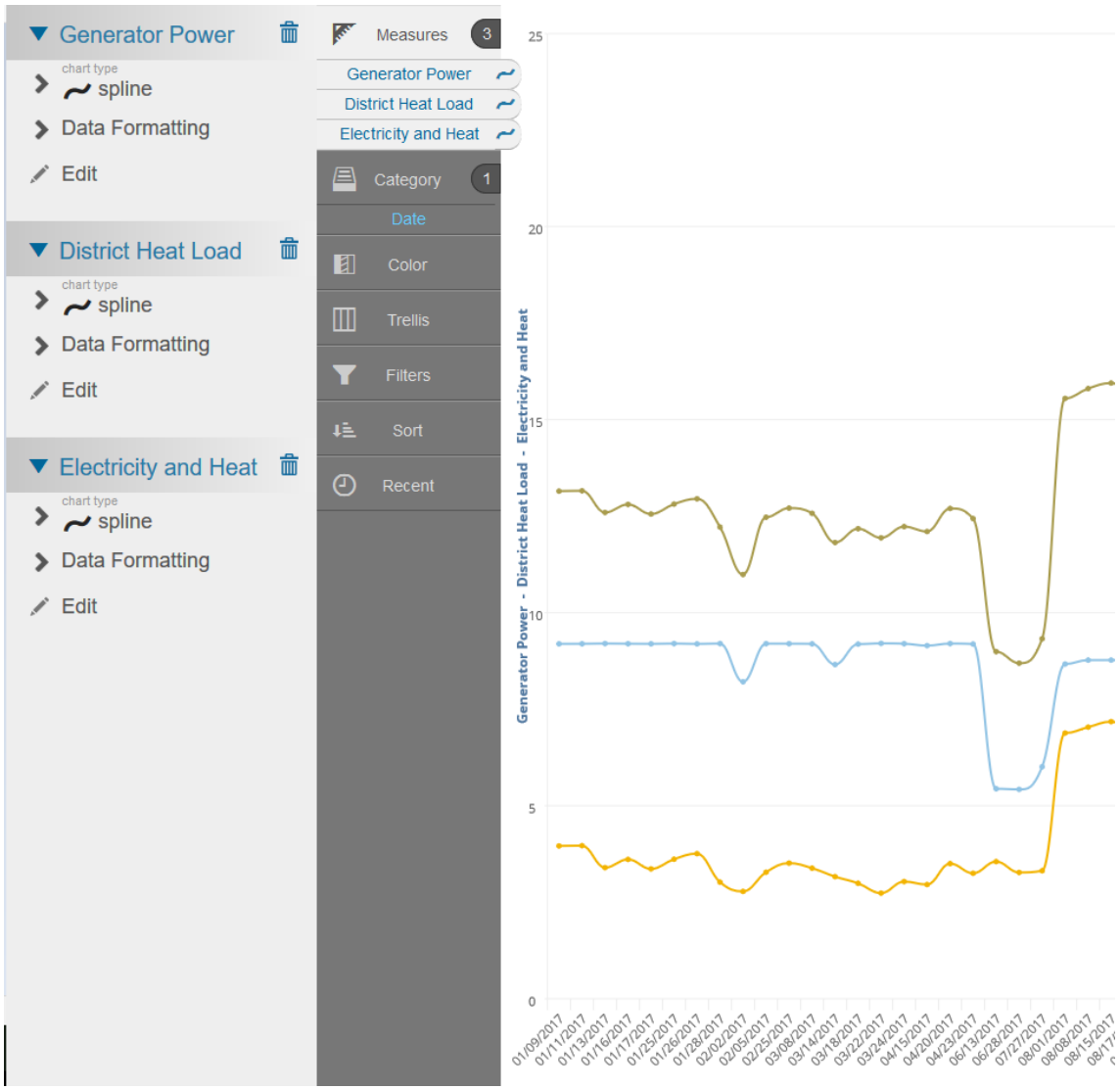


Figure 28. *Visualizer view with a sum signal included.*

After the trend is configured with the desired information, it can be added into the dashboard. Dashboards are the views presented to the customers and are made of elements called *dashlets*. Dashlets can include sliders, trends, indicators and other widgets the user can interact with and view the desired information. Figure 29 shows the trend as a part of the dashboard. The dashboard layer can be included with filters, which automatically retrieve data matching the set criteria, such as a specific timeframe, for all dashlets on the dashboard.

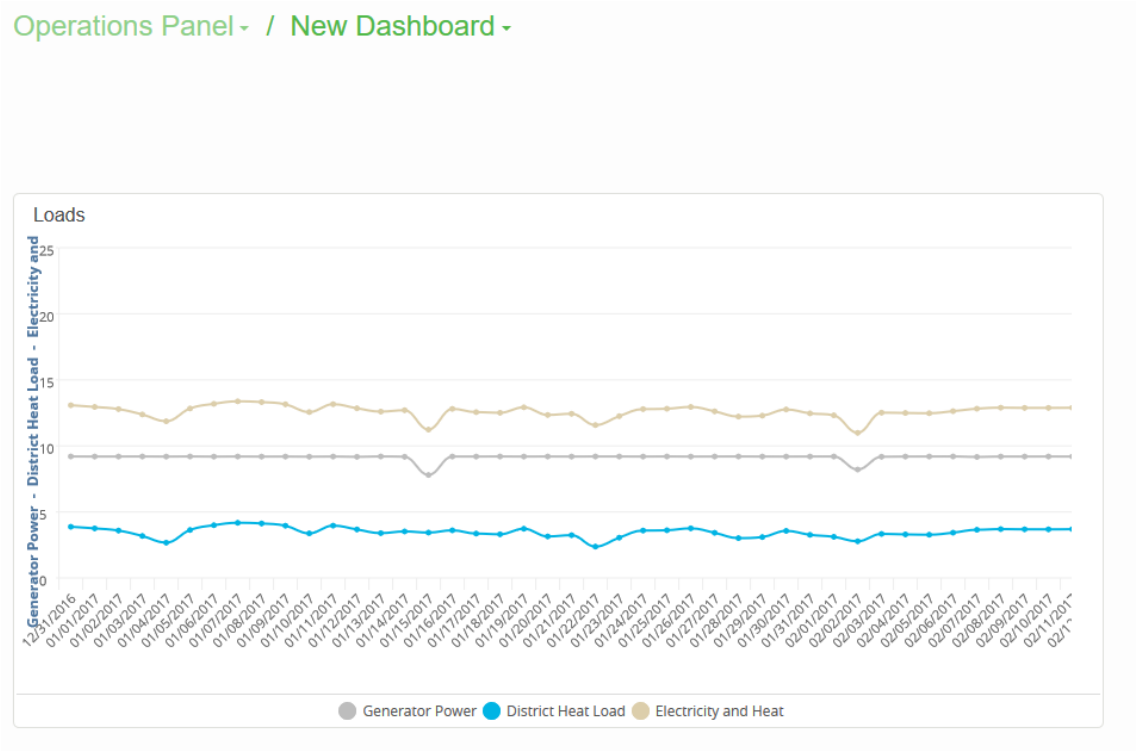


Figure 29. *The trend as an integrated dashlet on the dashboard.*

The final dashboards are made of collections of these dashlets, and the full site the customer can access contains multiple linked dashboards.

3.5 Calculation Procedure

As seen in the previous chapter, it is possible to perform simple arithmetic with BQL. Birst even has an integrated KPI calculation tool, which can present aggregates in a single figure based on BQL queries. Many KPIs defined earlier are actually quite simple, requiring mostly simple arithmetic or not even that. However, calculating the KPIs with Birst has two major drawbacks:

- The BQL is not suited for complex logical comparisons or other calculations requiring more flexibility
- The KPI values are not stored anywhere

Due to the first point, calculating some of the more complex KPIs is impossible with Birst alone. The second point is problematic because the values presented on the dashlets are calculated from the raw data during visualization. If the customer wants to purge his data from the cloud, the KPIs would be lost as well. This is unfortunate, since those KPIs are also used by Valmet for internal quality control and performance benchmarking, and as derived calculations can legally be contained.

For these reasons, a separate method for KPI calculation was devised. The AWS platform contains an integrated serverless computing platform named *Lambda*, which can run code in response to predefined triggers called *events*. These triggers include events such as new data arriving in a specified container or pulses in specific time intervals, which are ideal triggers for calculating KPIs. Lambda can natively support many programming languages, of which Python was selected. Python has already been in use at Valmet for similar projects, most notably due to having many immediately useful libraries.

Before any programming was done, a plan for the system operation was laid out. Besides just calculating the KPI values, the system had to be robust enough to handle all sorts of errors and unexpected events. The first thing to think through was to visualize what sort of information would be needed for the system. In practice this meant designing tables for Redshift, which would hold different layers of information.

3.5.1 Auxiliary Tables

The first table to be devised was a simple table of two columns: a name of the KPI and the Valmet standard unit of that KPI. This table simply lists all the unique KPIs that have ever been defined and contains a standard unit for them. This table is used by the scripts to check if a KPI intended for calculation even exists, and to extract the standard unit. Important to note is that if for some reason the standard unit would want to be changed, it only needs to be done here, and the script re-calculates the old values to the new unit in the next process cycle.

kpi_name	kpi_unit
Lambda	-
SupportFuelUsa...	MW

Figure 30. *A portion of the KPI list table.*

The second table deemed necessary defines the KPI calculation variables and contains three columns: a KPI name column, a VII variable name column and a VII variable unit column. VII variables are a way to handle the differences between plants. Each KPI is defined by general VII variables and is always calculated the same way, regardless of the plant layout. VII variables, on the other hand, are calculated from the information available from the plant and are thus plant-specific. For example, the definition of the KPI 'Primary Air Ratio' is always the same, total primary air / (total primary air + total secondary air), but different-sized plants have differing amounts of air ducts and thus signals to sum to form the 'total' values. This way, if the KPI definitions need to be changed, it only needs to be done once, not for every plant it was defined for. The table links the KPIs to the variables used in their calculation, as well as records the unit the KPI calculation expects the variables in.

kpi_name	vii_variable_name	var_unit
SupportFuelUsage	TotalLoadBurnerPower	MW
SupportFuelUsage	TotalStartupBurnerPower	MW
Lambda	FlueGasO2	%

Figure 31. *A portion of the KPI definition table.*

The third table, also known as the sites table, records all the KPIs that have been assigned to some site. It has four columns: a KPI name column, a mill id column, a line id column and a site KPI unit column. Mill id and line id together specify one unique *site*, a cohesive unit for which calculations can be performed. For example, some plants may have two boilers, in which case they would have the same mill id but different line id, and both would have their own set of KPIs. The site unit tells the desired end unit of the calculation. The customer might want to view the data in kilowatts while the standard unit is in megawatts, so at some point a conversion is needed. This is also crucial for customers who use some other unit system than the metric system.

kpi_name	millid	linei	site_kpi_unit
Lambda	402596	002	-
SupportFuelUsage	402596	002	MW

Figure 32. *A portion of the KPI sites table.*

VII variables are plant-specific and are calculated from whatever information is available at the current site. The variables table binds the customer tags to a VII variable on the same site. It has five columns: the VII variable name column, mill id column, line id column, customer tag name column and required unit column. Each row binds one customer tag on one site to one VII variable. (Of course, the same signal can be used in multiple variables as well.) The required unit column records the unit *in which the VII variable calculation expects it.* This does not mean that the customer tag is necessarily stored in that unit, but it tells the script that if the unit is different, a conversion has to be performed.

vii_variable_name	milli	line	customer_tag	req_unit
FlueGasO2	402596	002	1HNA10CQ302:av	%
TotalLoadBurnerPower	241660	001	3HHE38DU901:av	MW
FlueGasO2	402596	002	1HNA10CQ303:av	%
TotalStartupBurnerPower	241660	001	3HYA10DU913:av	MW
TotalStartupBurnerPower	495867	006	4HHA22CE201:me	MW

Figure 33. *A portion of the VII variables table.*

The final table to be devised was the result table. It has seven columns: mill id, line id, KPI name, KPI timestamp, KPI value, KPI unit and calculation timestamp. Each row of data tells the site the result was calculated for, the KPI and its value. The KPI timestamp is the timestamp of the *data* from which it has been calculated, while calculation timestamp is the time when the value has been calculated. This was deemed important for manageability: if at some point some KPI definition would have to be altered, or unit

changed, etc., it helps when there is a timestamp of the calculation to determine if the values in the database are already updated. The KPI unit is strictly speaking redundant information, as it is already included in the list table, but it was included to make the result tables easier to read.

Besides the primary result table, another almost identical table was also created for non-standard calculation results. During execution, the script would normally calculate the KPI values to standard units and insert them in the primary result table. If the script then detects an inconsistency in the standard and site units, it performs the conversion and sends the results in the nonstandard table as well. This might sound redundant, but it accomplishes two things: the values are always stored in the form the customer wants to see them, which makes visualization faster and easier (since there is no need for conversion in Birst), and the values are also readily accessible by Valmet experts, who might not want to work with non-SI units.

millid	linei	kpi_name	kpi_timestamp	kpi_val	k	calculation_timesta
402596	002	SupportFuelUsage	2017-07-02 03:30:00	13,50271800	MW	2018-07-27 08:08:37
402596	002	SupportFuelUsage	2017-07-02 03:31:00	13,68341000	MW	2018-07-27 08:08:37
402596	002	SupportFuelUsage	2017-07-02 03:32:00	13,93344100	MW	2018-07-27 08:08:37
402596	002	SupportFuelUsage	2017-07-02 03:33:00	14,04318500	MW	2018-07-27 08:08:37
402596	002	SupportFuelUsage	2017-07-02 03:34:00	14,19329400	MW	2018-07-27 08:08:37

Figure 34. *A portion of the standard result table.*

3.5.2 Script Overview

When the script is run, it first establishes connection to Redshift. If the connection fails, the script is aborted outright. After that, the script attempts to import a Python module containing the KPI calculation definitions. These are located in a separate file to differentiate between the main script logic and the actual calculation logic. This should make maintaining code easier, since ideally after the main script is deemed complete, only the KPI definition file would have to be altered in the future.

After the KPI module has been imported, the script performs a query to the Redshift database asking for the contents of the KPI sites table. In a fully automated mode, the whole list is iterated and every KPI for every plant calculated. The table is returned sorted by mill id, then line id and finally KPI name, meaning that calculation is performed site-by-site. The script picks the topmost row and starts to perform calculations row by row.

For any KPI sites row, the script first attempts to import another Python file, containing the calculation methods for the VII variables of that site. Each file is named by the same structure, “millid_lineid_vars”, and contains functions named after the KPIs for which they produce results, e.g. “SupportFuelUsage”. This distributed model accomplishes two things. First of all, it is much easier to manage code when the logic for each site is located

its own container, allowing multiple people to create their own implementations. Secondly, it allows easy handling of the KPI calculation in a for-loop in the main script. Since the correct module is always imported for a site, and the Python files contain identically named functions, a simple if-fork for the possible KPIs is enough to find the correct calculation procedure. Otherwise every possible site-KPI-pair would have to be coded separately. An overview of how the distributed model works is presented in Figure 35.

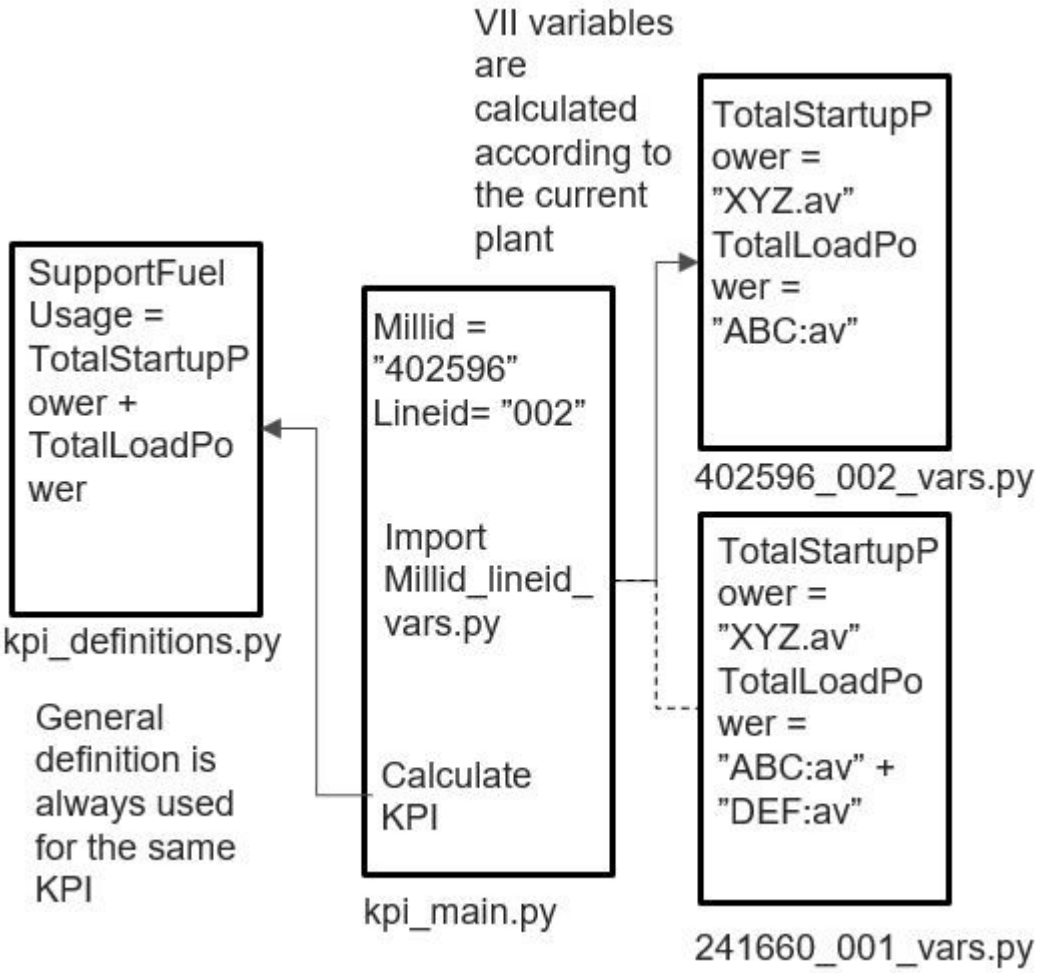


Figure 35. *Overview of the script import logic.*

When the variable module is imported, the script gets the standard unit of the current KPI from Redshift. Then the calculation time interval is defined. By default, the script fetches the latest calculation timestamp for the current KPI and calculates from that time to current time. If the latest timestamp does not exist, however, the script calculates the last 5 minutes instead. After that the script gets the Valmet compound tags for all the signals involved in the calculation. It first uses the KPI definition table to check which VII variables belong to the KPI at hand. Then it uses the VII variables table to get the customer names of the required signals at the current site, and finally fetches the corresponding Valmet compound tags from the metadata table. This is because the actual process measurement values are logged by compound tag, not by customer tag. Lastly, the script fetches the units for all the relevant signals.

The next step is to fetch the actual process values for the relevant tags. The result table has a timestamp column and as many additional columns as there were signals. Each row contains the signal values for all signals for that timestamp. The program can't handle null values, so the query simply omits any row where even a single signal would have them. Also, even if the script is set to retrieve data from a select time interval, it can only ever return data from those timestamps for which all signals included some value, meaning that if for a five-minute interval only three minutes contain all values, only those three are returned. The script performs quick comparison on the retrieved units from the metadata table as well as the VII variable table, and if unit inconsistencies are found, it attempts to convert the values to the unit specified in the required unit column.

At this point, the script uses the imported variable module to calculate the VII variables from the raw data. The module performs some error checking of its own, ensuring that the data is of the correct type, does not contain illogical values (such as a negative power) and that a correct number of signals were provided for the function. The function then performs the calculation according to its implementation and returns the VII variable results to the main script, with a timestamp column and as many columns as there were VII variables. Each row represents the values of the VII variables for that timestamp.

The next step has the script to probe the result table for the defined time interval. This query returns a table of any values already existing for that time interval, KPI and plant. This information is used later. After this the actual KPI values are calculated in the KPI definition module based on the VII variable results. The result matrix has two columns, one for timestamps and the other for the corresponding KPI calculation results.

Finally, the results are written back to the Redshift database. In this step, the script also uses the existing records data to determine whether it should insert the value as a new value or update the old one. An important aspect during production was to ensure that the automated scripts cannot fill the database with thousands of duplicate values or values of ambiguous origin. Thus, the script will only insert new values if no old data exists. Additionally, if the site KPI unit was different to the standard one, the script performs an extra procedure where it converts the values to the site KPI unit, retrieves the old records from the nonstandard result table and applies the same insert-update -logic on that one as well. After this the script picks the next row from the KPI sites table and begins all over again, until all the KPIs have been calculated.

The script also includes some switches for different modes of operation. The update_toggle switch causes the script to only perform insertions when set to false. The smart_update switch, when set to true, will only update a value if the new value is beyond a set percentage threshold of the old one, or if the new unit is different from the old one. The take_backup switch causes the script to write any values set for update in a CSV file first, and the log_errors switch causes the script to generate an error log text file after running, detailing the reasons why some particular KPI calculation failed. Also, the script can be

supplied with manual inputs for the KPIs, sites and time interval, causing it to include only the manual inputs in its queries.

3.5.3 Lambda Integration and Scheduling

The actual programming is done on a local machine running a Python development kit, where the developer can run, test and tweak the code. After the code is deemed ready, it must then be uploaded to AWS Lambda and configured to work in a cloud environment. The first step is to include all source code and relevant libraries in a single ZIP file. In AWS Lambda, a function is then created to contain the incoming package. For the function, a *role* must be specified to limit the online resources which it can access, e.g. only allowing access to the Redshift database. Lambda supports several languages as runtime environments, which can be selected here. Figure 36 shows the Create Function view.

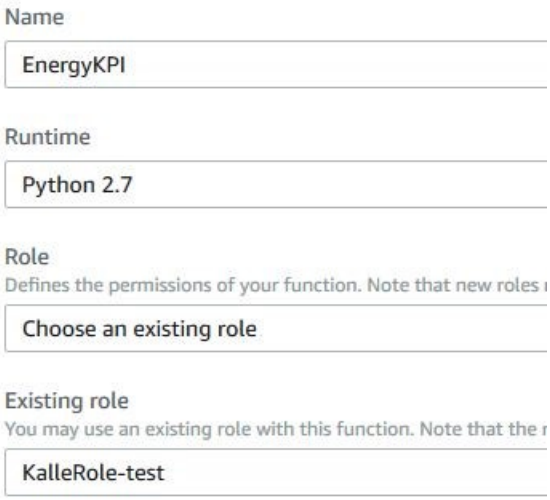


Figure 36. *AWS Lambda, Create Function view.*

After the function is created, it must be configured. The first thing to do is to upload the ZIP file containing the actual code. Lambda automatically extracts and deploys the code upon arrival based on the selected runtime environment. If the program is small in size, the code can be edited online in the function view, but for larger programs, any edits must be made locally and the package must then be re-uploaded. A *handler* must also be specified. This is the name of the function inside the deployment package that is called upon when the package is run. The function is configured with a maximum amount of memory it can use inside the cloud, as well as a timeout limit, which causes Lambda to terminate any programs that use more than the specified time to finish. A virtual private cloud, a logically isolated segment of AWS cloud, is set up for Valmet developers, where they have full control over the networking environment, for example to create subnets and limit Internet access. The Lambda function is configured to the private cloud, along with any subnets and security groups. Overview of the display is seen in Figure 37.

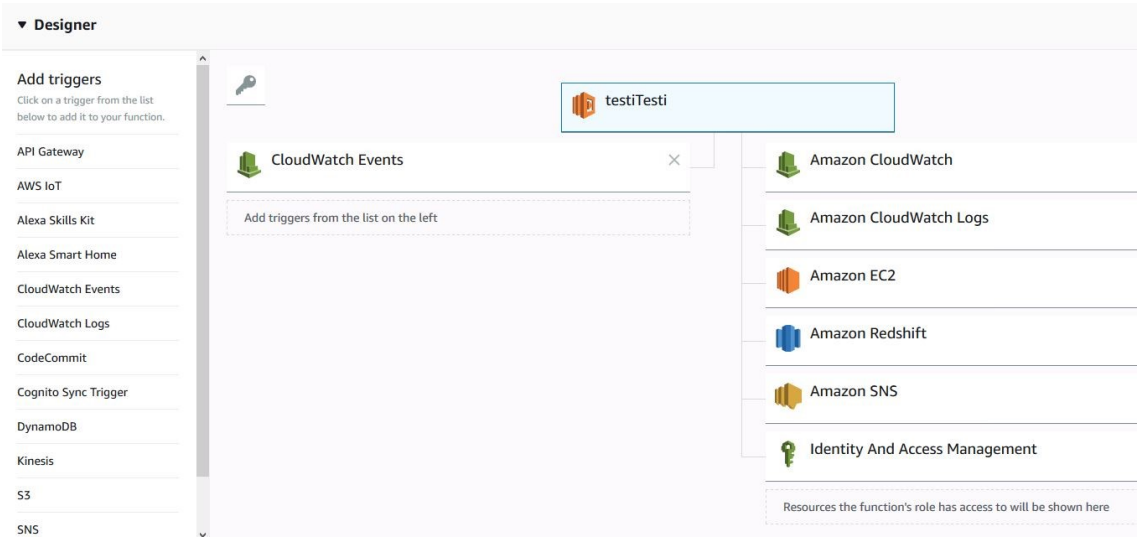


Figure 37. *Overview of the Lambda deployment display. testiTesti is the deployment package, below it are trigger sources on the lower left and available resources on the lower right.*

After the function is deployed, it can be run manually with preset inputs in the Lambda function view. However, the true power of the cloud is not achieved until the scripts can be set to run automatically. For this, the AWS *CloudWatch* service can be used. CloudWatch includes an event handling logic, which allows developers to use CloudWatch events as triggers for running their functions. Events can be complex logical rules, but for KPI calculation a simple scheduler is ideal. At first, it might seem logical to have the script react to new data arriving in the cloud, but the script is set to calculate every KPI in every plant automatically. Even if all the plants connected to VII do send their data packages in exactly 5-minute intervals, the different plants might still be out of phase with each other, meaning plant A will send the data now, plant B 7 seconds after that, plant C 2 seconds after that, etc. The script must have enough time to run from start to finish before it can be told to run again, so for this reason the script is scheduled with a 10-minute pulse independent of the plant data flow. Example of the scheduling view is seen in Figure 38.

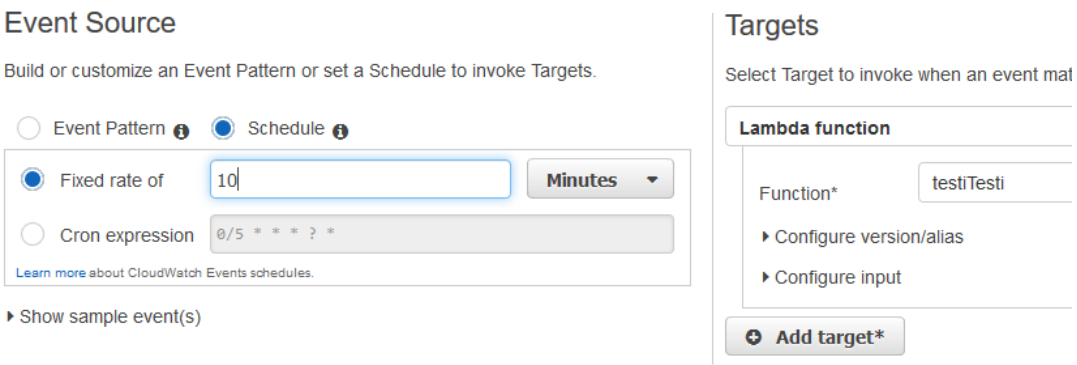


Figure 38. *Configuring a CloudWatch trigger for a 10-minute interval.*

4. REVIEWING THE SYSTEM

After the data pipeline was implemented and KPI calculation scripts written, extensive testing of the system began. The first attempts were mostly for debugging, since having actual process data revealed logical errors which randomly generated test data could not catch. One such instance was the inclusion of boiler load data in KPI value assessment. The script performed the calculations as intended, but the resulting numerical values were not restricted in any way. This lead to situations like division operations with the divisor approaching zero, such as the case with sand consumptions during periods of boiler shutdown. As long as the value was above zero, even if slightly, the calculation was performed, resulting in very large numbers. Such cases were then handled by comparing the results to the boiler load: if the boiler load was below a set threshold, the boiler was probably on shutdown, and the KPI value was set to be zero. It should be noted that people who read the data tables should have an understanding of the process and information about load conditions, so they can make their own judgements about the correctness of the data. After all, a value of zero could be generated by the script naturally as well, so only with data that offers a context to the results can a process engineer assess the boiler conditions. What this type of filtering achieves, however, is that extremely high values won't skew any hour-, day- or month-averaged aggregates that might be generated from the results.

The system was robust enough not to crash at exceptions. If a KPI calculation failed, the script output an error message detailing the cause, but continued calculation for the rest of the set. This is important when the script is running automatically in the cloud, since the abortion of the program mid-way would mean that the next time the script would have to calculate the values for the remaining half for a time interval twice as long. If this continued for long enough, the remaining half might eventually take so long to calculate that the script would timeout.

4.1 Results

To test the system, a single batch of data from the pilot site was extracted. This data contained varying load situations and a few shutdowns, giving insight to the performance of the system under different conditions. Live data extraction and automatic KPI calculation were not tested here. To showcase the results, a total of six KPIs were calculated from the raw process data using the script.

The first KPI to calculate was steam load percentage, a simple division of two values, one of which was constant. Results are presented in Figure 39.

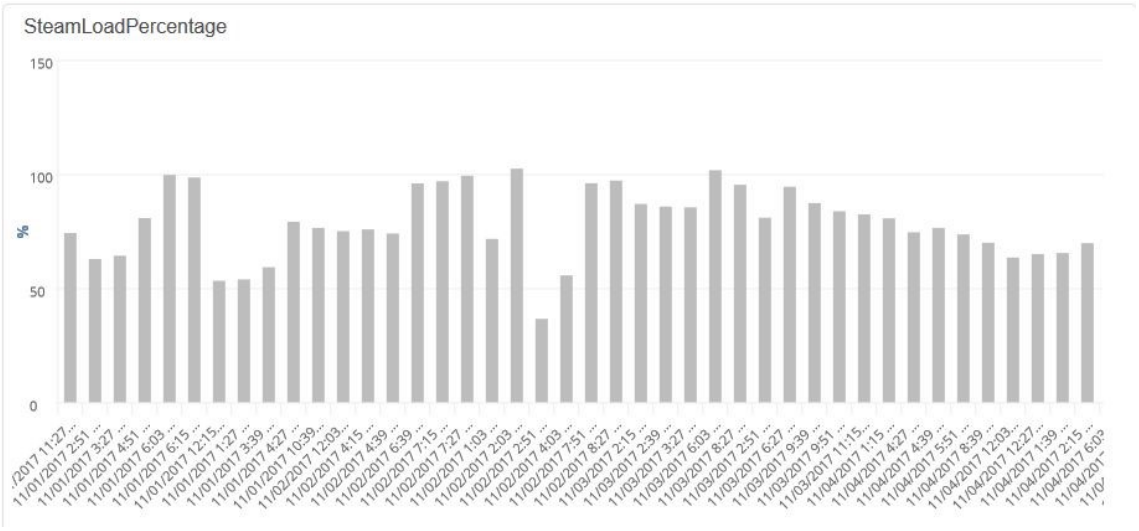


Figure 39. *Steam load percentage, calculation results visualized in Birst.*

Study of the results showed that the values varied between 0 and 100 %, as expected. Birst performs some aggregation depending on the selected amount of data points shown and the time interval, but these can be adjusted by the user. Rest of the values can be seen by scrolling the dashlet to the right.

Figure 40 contains values for the excess air coefficient:

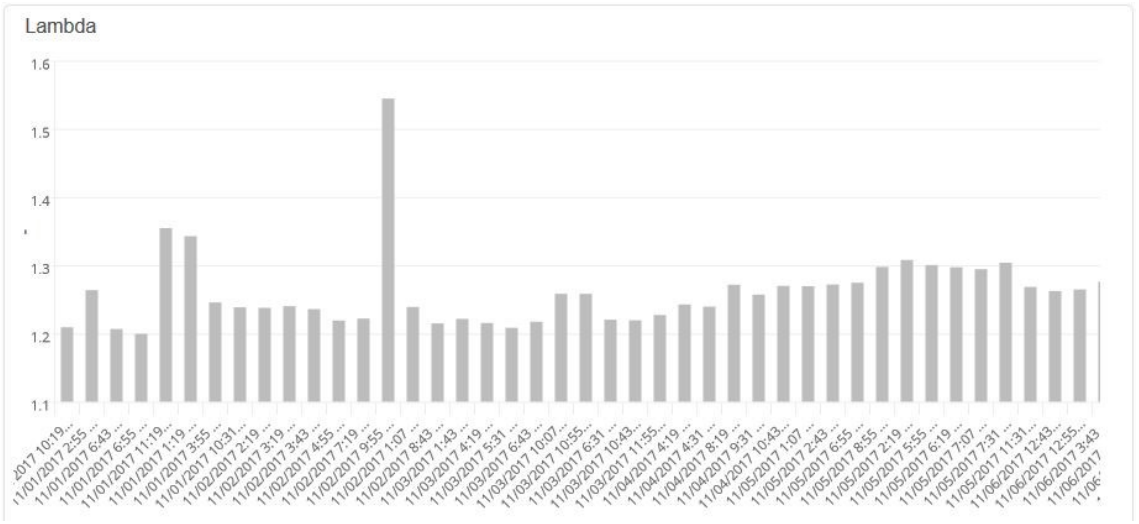


Figure 40. *Total air coefficient, visualized in Birst.*

The results showed that most of the time the average air coefficient was around the design point, and spikes only appeared during low load conditions.

Figure 41 shows the own power consumption:

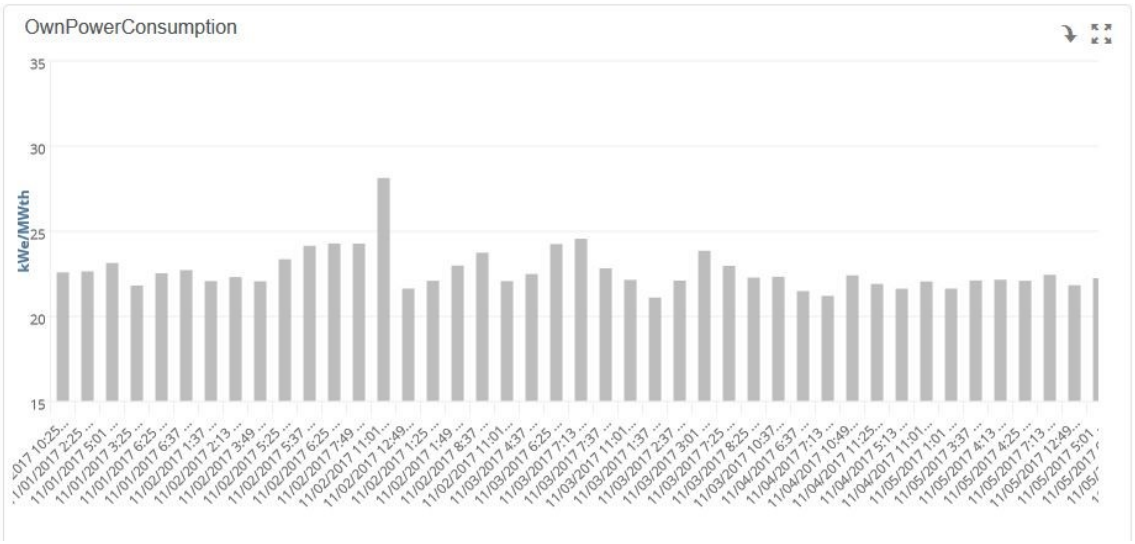


Figure 41. *Own power consumption, visualized in Birst*

Here Birst displays its ability to perform simple arithmetic during visualization. The actual values are stored as MW_e/MW_{th}, and this dashlet was configured to multiply the values by a factor of 1 000, giving kW_e/MW_{th} instead.

Figure 42 shows the sand consumption:

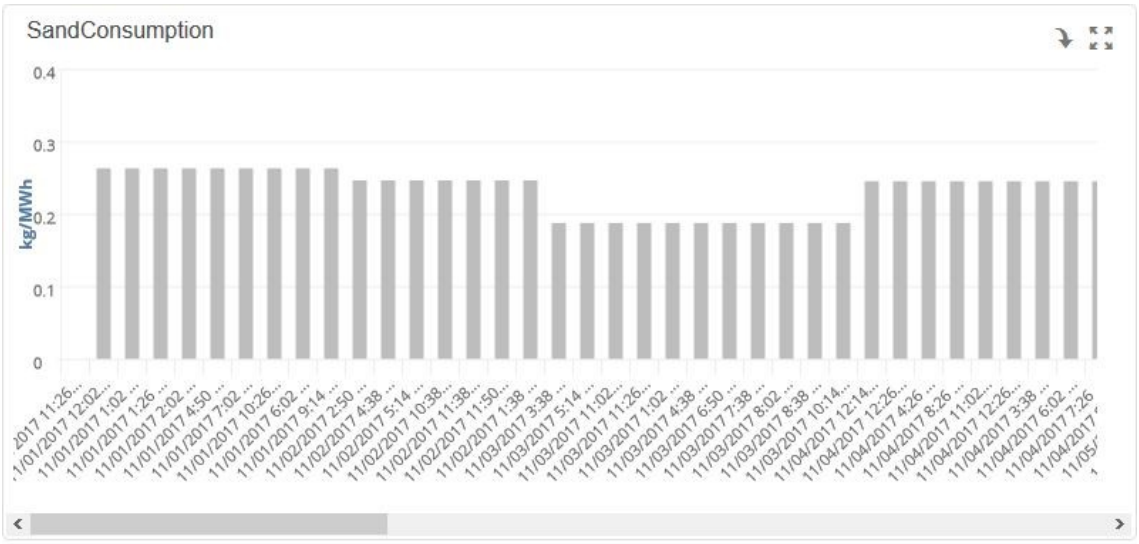


Figure 42. *Sand consumption, visualized in Birst.*

Since the values are day-averaged internally (i.e. the minute values between 0:00 and 23:59 for each day are all the same), Birst aggregation also creates discrete steps. This KPI might later be better to store as day-average value from the start. The values themselves, however, were sensible.

Figure 43 shows sand consumption as a 24-hour average:

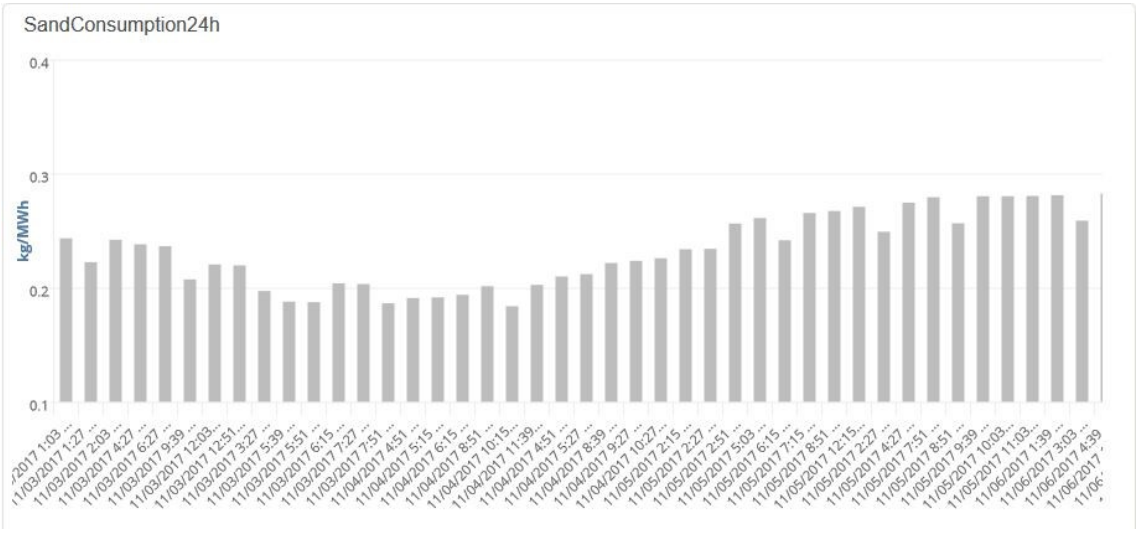


Figure 43. *Sand consumption, 24h average, visualized in Birst.*

This is a variation of the last KPI, where instead of a daily average, a rolling average is employed. Here the values change each minute, so a minute-value storage is also mandatory.

Figure 44 shows the results for the primary air ratio:

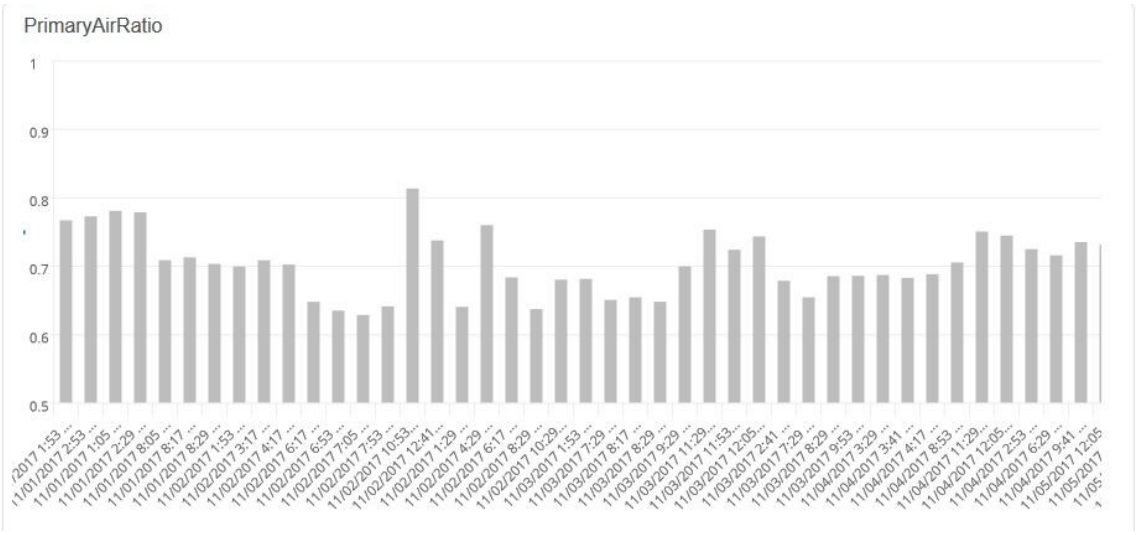


Figure 44. *Primary air ratio, visualized in Birst.*

The values are in fractions, but they could also be presented as percentages with Birst. The data was consistent throughout the time interval.

Additionally, a seventh KPI was calculated. This was the ID fan vibration time out of range. However, the process data did not have any instances of the vibration measurement actually being over the safe range, so the result table was flat zero.

The time series shown above are not the only way to study and visualize the results. The data itself resides in the database as minute-averaged values. Simply plotting by the associated timestamp shows the values as a time series, representing the way the values have changed over time. However, if sorted and plotted by value in descending order instead, a duration curve would be produced. Duration curves show the amount of time the plotted value has been over a certain limit. A typical application of a duration curve would be steam load percentage, shown in Figure 45.

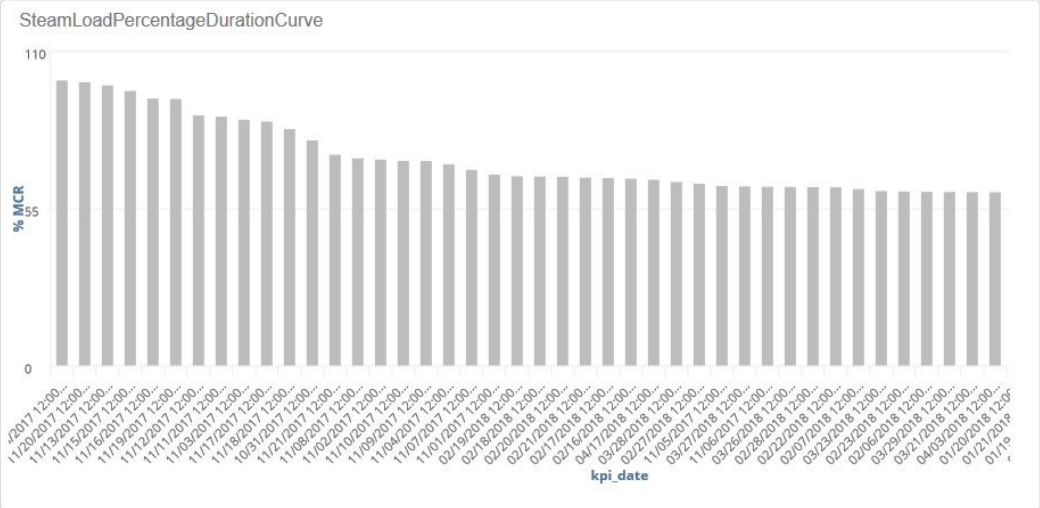


Figure 45. *Steam load percentage as a duration curve. The rest of the curve can be seen by scrolling to the right.*

Another useful way to visualize the data is to use an XY-scatter. Scatters plot the values of one variable against the corresponding values of another variable. This is a powerful way to quickly find and showcase correlations, dependencies and trends between two signals. For example, plotting the own power consumption against steam load reveals the electricity usage of the plant based on load conditions, as shown in Figure 46.

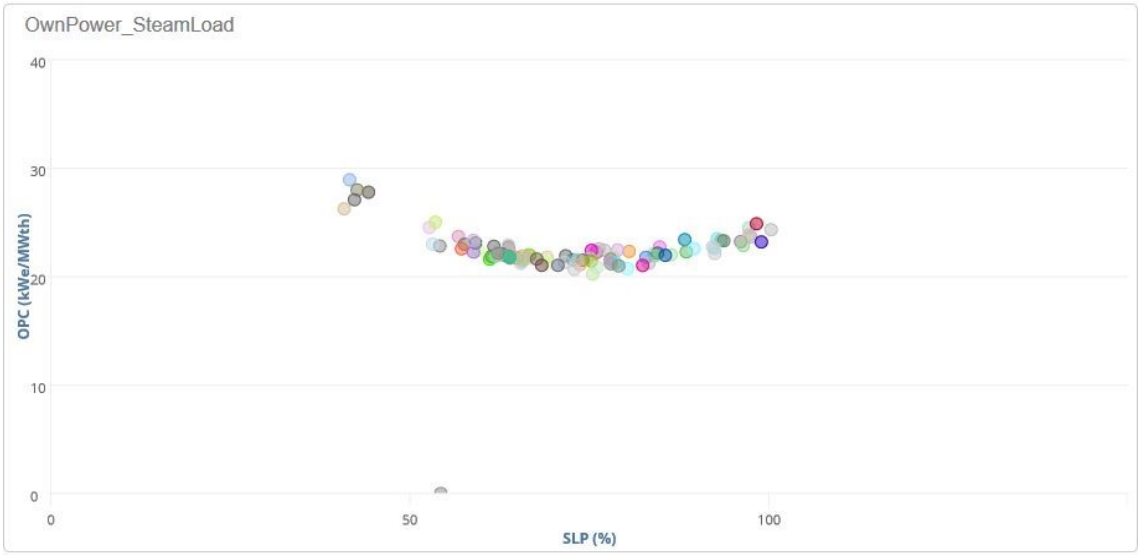


Figure 46. *Own power consumption plotted against steam load. The lowest electricity consumption seems to be at around 75 % load.*

4.2 Speed

Calculation speed is of crucial importance in assessing the usefulness of the script, and during development was the single most common reason for revisions. Oftentimes the first implementation would work reasonably well with a small set of data, but the calculation times would quickly rise exponentially along with the amount of data points. Three largest contributors to process time can be categorized as follows:

- Data retrieval from Redshift
- Data structure management inside the Python script
- Calculation result export to Redshift

To function as intended, the script must probe various kinds of information from the AWS database. The amount required depends on the KPI at hand as well as the defined time interval. Each calculation loop contains the same steps – data retrieval, unit checking, calculations etc. – but the speed depends on the requirements of the KPI at hand. The first iteration of the script would fetch the process data as $1 + x$ column tables, where x was the number of signals required. This worked well until a KPI calculation required more than 15 signals at once. Internally, such queries were achieved with x amount of database *joins*, which cumulatively slowed down as the number of signals went up, until finally the processing time shot up to tens of minutes. This was handled by breaking down the fetch query to parts and finally joining the data tables inside Python.

Another consideration is the actual time interval and thus the number of data points. Each iteration of the script has to fetch the process data for the set time interval, then the existing records for updating and boiler load data for the same interval. The time it takes to do all this is different when calculating over a 5-minute interval (5 data points) and a full

year (525 600 data points). Query times also vary each iteration, because the Redshift database has many other concurrent users, causing variable loads during iterations.

Data structure management here refers to the implemented methods of handling the data and performing the necessary actions inside Python. Data structures in this case are often matrices, containing anything between 2 to 20 columns and 5 to 500 000 rows, and multiple such tables have to be passed to functions as arguments, iterated through or manipulated somehow. The single largest issue with data structures was with insertion of new data. The Python library used for handling 2D arrays, *NumPy*, supports adding rows or columns to the bottom of the matrix and worked fantastically with limited data sets. However, with more data, the time it took to create arrays spiked. The reason for this was that internally NumPy re-creates the entire array with the added row at the bottom, causing each iteration to take slightly more time, which was evident with huge data sets. This was amended by creating Python lists for each column and appending the elements to the end of the lists during each iteration. Appending an element to the end of a Python list is an operation the time of which is independent of the current size of the list, and after the lists were ready, they were stacked into a final 2D array with NumPy. This simple change brought the calculation times from hours to minutes, showcased in Figure 47.

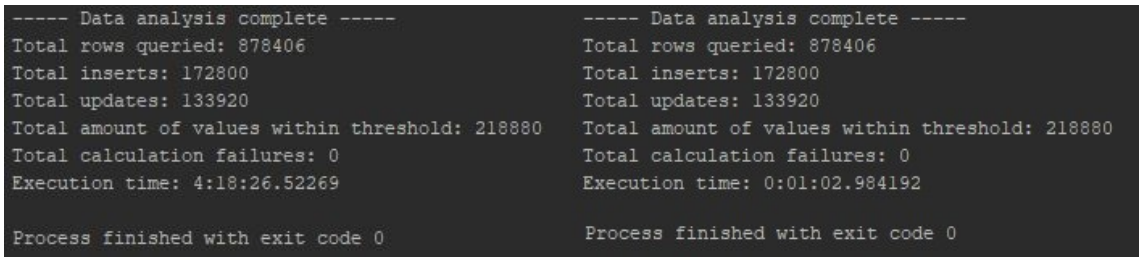


Figure 47. *Before and after pictures of the script runtime when implementing new way to create arrays.*

Another point with arrays is that their manipulation and examination often requires nested for-loops, iterating either a subarray or the whole array. For-loops are somewhat inefficient in dealing with large arrays, but after examining the runtimes, they were deemed sufficiently fast in comparison to other, more pressing issues.

Data export is basically the data retrieval reversed. However, the methodology of this has a great impact on the script speed. In the first version, each result table row was inserted or updated separately. This was wildly inefficient due to the database doing all the associated background work each time. Especially the update of a large number of rows took long, because the database had to single out the one row to be updated in the result table every time. Insertion was simple enough to fix by having the script generate a query with thousands of rows in it, but there was no such method for updating thousands of rows at the same time. The fix was to create an auxiliary table for the update rows, insert them there and run an update join inside the database. This made updating as fast as inserting,

plus the join time, which was a fraction of the insertion time. The auxiliary table is then emptied after each update.

The speed of the script might be a bottleneck in the future, when there are 300 KPIs to calculate, 200 sites to go through and only 5 minutes of time before AWS Lambda timeouts the script. Even if the time intervals are 5 to 10 minutes, the many consecutive queries, insertions and iterations will take time. In that case, the script might need to be broken down for each plant or for each KPI.

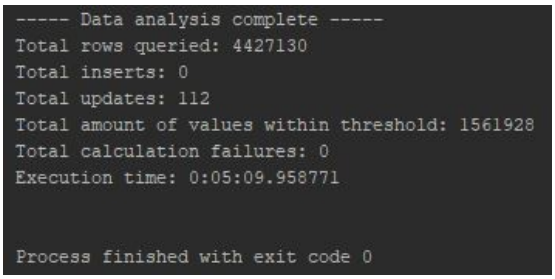


Figure 48. *The time it currently takes for the script to calculate 7 KPIs. Data range is about six months.*

Figure 48 depicts the current performance of the script, with 7 KPIs calculated from a six-month interval. The time is more than the AWS Lambda timeout limit, but in the cloud, the script will only have to calculate 5-10 minutes of data for each KPI, making calculation much faster.

4.3 Accuracy

All the data generated by the scripts and stored in the cloud is of no value unless it is validated. For any given calculation, there are multiple possible sources of errors on the path to the cloud:

- Sensor accuracy in the process
- Dynamics in the process
- Possible number truncation or rounding inside the DCS
- The minute-averaging function
- Possible number truncation or rounding in the cloud
- Possible number truncation or rounding in the Python script
- Logical errors inside the calculations

The first point to consider is the actual source of the raw process data, the measuring equipment. Even the best sensors have tolerance ranges, varying between the type of sensor as well as the measured variable. For example, an accurate temperature sensor such as a resistance temperature detector (RTD) can have an uncertainty of +/- 0.67 °C, while a common thermocouple can have as high as +/- 2.2 °C in the same temperature range [89]. The reliability of a sensor is divided in two groups: the *accuracy*, or how close the

measured values correspond to the true value, and *precision,* the ability to reproduce the same measurement results under similar conditions [90]. Sensors commonly used in process industry include temperature, pressure, level and flow sensors, with each variable having their own range of measuring equipment, which in turn have different tolerances.

The cloud does not inherently receive any information about the type, calibration or conditions of the sensor, so the values can only be assessed with other data providing context. The most notable problem comes from the fact that in the KPI calculations, those values, which might be anything between 0.1 and 20 % off the true value, are used in calculations such as multiplying by a factor of 1 000, or multiplying two already erroneous values together. This propagates the individual error in the raw data to the calculation sub-results and in turn the final results. As such, even if the calculation procedure is implemented correctly and the raw data is sensible, the results should still be verified to see if they make sense for the plant at hand. It should be noted, however, that the actual plant itself relies on those same sensor measurements, meaning that there has most likely been much insight into their accuracy already. Nevertheless, VII might also help identify plants where measured values do not quite correspond to the expected ones, or at least spot clearly miscalibrated sensors.

The next major source of possible errors is the minute-averaging function. Any values sent to the cloud for analysis and further calculation are not instantaneous measurements, but aggregates of a one minute interval, compiled in the INFO computer. In an industrial process such as steam generation a minute can be a long time. Aggregating collects information from the whole time period and only generates one value representing that time frame, but the end value might hide the actual fluctuations in the process, or be influenced by them. Instantaneous values are the exact measurement values at that time, but for a time interval as long as a minute, they hide too much information between them to justify simple interpolation. The values produced by the sensory equipment contain *noise*, random fluctuations in the value resulting from uncontrollable sources such as thermodynamic motion or turbulence, as well as any errors in the calibration or assembly of the device. Random noise deforms the true underlying signal value, which can be seen as erratic spikes around the expected value. The *signal-to-noise ratio*, SNR, represents the quality of a measurement, but requires some information about the noise itself. Aggregating improves the SNR by averaging out the random noise fluctuations. The minute-averaging function might also contain logical errors as well, but that is not the subject of this thesis. [91, 92]

Another point to consider is that some events in processes are not instantaneous. Boiler processes have a *time lag* in many cause-effect -relationships, meaning that the effects of some control procedure are not immediately visible. A pivotal example would be a change in boiler load. Even if the fuel load is increased, the actual observed effects on the main steam temperature and flow appear only after a delay. The time lag is greatly dependent

on the nature of the process and its implementation. Process dynamics should be considered when designing KPIs, since the values might not be representative of the system during the delay time. For example, if boiler efficiency were calculated with instantaneous values, the apparent efficiency would be lower during the time between increasing the fuel load and observing the effects in the main steam. Aggregating alleviates this problem somewhat, since it includes data from a range of time, instead of a specific moment.

Number truncation and rounding at different stages of the pipeline can be problematic if the number of significant figures falls too low at some point. The way the numbers are handled varies between systems, for they can be presented as 64-bit double-precision numbers in the first step and as 32-bit floating-point numbers in the next. The least accurate step along the way takes precedence, which is most probably the Redshift data vault. The data there is stored in columns with a set format, one of which is the NUMERIC type. Numeric format is given two parameters: the total number of significant figures stored, and the number of decimal places stored. If the raw process data is stored in precision of 8 decimal places, the accuracy of any values before that is irrelevant, and any calculations performed based on that data can only be accurate to the 8^{th} decimal.

Lastly, assuming all the data to this point is as accurate as possible, the final source of errors is the calculation script itself. Syntax and semantic errors are easy enough to spot, since interpreters will catch those, but when everything runs smooth and some values are generated, it raises the question if the values are correct. Generally speaking, the person who implements the calculation for a given KPI or plant variable set takes responsibility that the calculation results from his or her step are correct, assuming the data provided for their calculation is correct. This does not prevent them from performing additional data quality checks, such as checking for values below zero when it would make no sense. But after the script produces results, it would be a good practice to check manually if the values are correct. This would most probably mean calculating a portion of the data in another program, such as Excel, by hand, and comparing the results. Finally, the opinion of a process-related expert would be the last stamp of approval for the calculation: if the same values are generated both by the script and the manual calculation, but the results don’t make sense, it might be that there is an error in the implementation logic.

The correctness of each of the above six KPIs was verified by calculating the same KPIs by hand in Excel. This proved fruitful, because some minor errors were caught, which would have been impossible to spot among otherwise good results. The results were then validated with process experts to see that the results were also sensible. Source sensor accuracy is not something the VII can affect, so the system will just have to trust that the values it receives are inside the set tolerance bounds.

5. DISCUSSION

The transformation of a company dealing mostly with traditional industry into a digitalized service provider is a huge change. The company will need human resources with all-new skillsets, and the existing employees will need to embrace a different mindset. This chapter contains observations about possible challenges that an Industrial Internet project might encounter. Solutions are proposed when applicable.

5.1 Generic Naming

As mentioned previously, signal naming in the cloud is not a trivial issue. Each plant has its own, internal naming scheme, which may or may not be a KKS-derivative. Even within single plants can exist different production units, where older and newer models have different naming schemes. One of the highlights of the Industrial Internet is the ability to gather data from numerous plants to a central location. That way boiler performance can be assessed not only on a single unit level, but via benchmarking numerous different units against each other. The reasons for performance deviation could be assessed with the help of reference data from other plants. This Fleet Management system would show customers their own plants and the service provider the whole delivered base.

This, however, raises the problem with signal naming. Even a simple query such as showing the main steam production from two plants would require knowledge of the plants' naming schemes, which itself is not the information the person making the query would be interested in. It should be possible to search with a query in the form of "main steam production; plant 1; plant 2", and the system would then search plants 1 and 2 for their respective main steam production signals. This of course requires there to be some sort of common name attached to the plant-specific signal: the generic name, by which the search algorithm can recognize the relevant signals.

The logic behind the system is simple enough, the problem starts with the implementation. Even though the general characteristics of boilers remain the same, it does not mean the setups are identical: some might have measurements in slightly different places, others have multiple measurements at the same place and still others not include it at all. Furthermore, many processes include parallel subprocess lines with their own measurements. Both the naming and numbering of the signals should be logical enough as not to warrant P&ID investigation.

Another aspect is the scope of the project. For this project alone some 1 700 relevant signals were identified in a single plant, 700 of which were physical measurements. Naming each one uniquely would have been a tremendous amount of work for a single person,

and in hindsight such a project probably even shouldn't be done alone, but in a dedicated project with the help of process experts.

In order to simplify the naming process, the generic names should be present on the schematics from the start. The best place to implement this would be the process engineering tool used to design the P&I, INT and CON diagrams. Namely whenever a component – sensor, motor, valve or other – is added to the schematics, it would also be given a descriptive name, such as Meas_main_steam_T or Pump_feedwater. After the diagram is finished, a list could be compiled for all the equipment with assigned extra names, which would effectively map the generic names with the plant-specific ones. This list could then be uploaded to the cloud as a lookup table for the queries. Main challenges with this method are:

- Should the numbering of measurements matter? If there are two parallel main steam lines, and both have their own flow measurements, should they be both named F_main_steam, or somehow separated via numbering? If numbering is applied, can the same query be used on plants with and without the numbering?
- How much difference between the description of a component and its actual implementation is allowed? If you expect to find a certain pressure measurement before a valve, and the actual plant has the closest equivalent after it, can you use the same name for it? Does it mean the same thing?

A compromise could be achieved by just adding the generic specifiers to the components during design phase and generating the mapped list. From this list is then manually compiled the plant-specific signal list, with numbering, specifiers and such added if required. This is quite possibly the first course of action until full automation can be employed.

5.2 Plant Instrumentation Readiness

Boilers around the world are currently made with an operational point of view. This is of course justified, since that is the main objective of the boiler, and all the data produced on the site has also stayed on the site. However, with the Industrial Internet, the data can be exported and analyzed to assess performance. This proves to be quite difficult if that data just does not exist.

In this context "instrumentation readiness" indicates the ability to extract meaningful information from the process. The main concern here are sensors which simply are not there. Consider for example a boiler with four economizer sections packed in two bundles. Quite possibly the only way to assess their performance is to look at the temperature measurements of flue gas or feedwater before and after the whole economizer set, since there are likely no intermediate measurements. This is even more apparent when designing Digital Twin applications, which are supposed to emulate boiler or component per-

formance based on historical measurement data. If detailed models about component performance is to be made, detailed data is also required, i.e. many more sensors than required in operation. This requires a change in both philosophy and mindset for the process engineers, who would previously only include sensors with immediate operational impact. The problem with data deficiency was apparent in the Own Power Consumption KPI, which includes the most notable electricity consumers. This is partly due to simplification and assuming that the rest of the machinery is irrelevant in comparison, but also due to the fact that not many other motors could tell their consumed wattage, either due to being fixed-speed motors or the power measurement missing from the inverter. The KPIs about power consumption could be made much more sophisticated if the data existed, such as comparing and then optimizing the power use of whole subsystems.

A solution to this would be to have internal meetings within the company by process engineers and boiler designers to discuss what sort of measurements should be added and always included in the delivery. However, it should be noted that the decision is not entirely for the boiler supplier to make, as the customer might object to needless costs. One remedy for this could be to offer to include the equipment for free, especially if the data is used mainly for internal product development, but this would of course require profitability assessment.

5.3 Raw vs. Refined Data

In this context, “raw” data refers to the immediate value the sensors measure and send to the DCS, and any additional calculations based on that value are considered refined (or pre-calculated) data. The DCS of any plant calculates refined values about the process based on the sensor input, and those values are used to operate the process. The calculations can include automatic conversion of measured flue gas components from wet to dry basis, or converting from gauge pressure to absolute pressure.

The main issue here is that it might not always be possible to see exactly how the DCS calculates those refined values from the sensor data. The problem will be especially profound on plants running a competitor DCS, where the Industrial Internet service provider has no access. The sensor values, however, will be the same physical measurements, so it could be possible to define a single calculation scheme in the cloud instead. Consider the enthalpy of water, for example. The plant DCS might use another correlation than the IAPWS-97, causing deviations from the expected values between plants. The pressure and temperature measurements for both plants could be converted to enthalpy with the same correlation in the cloud, possibly telling if the correlations between the cloud and the DCS are somehow off. This way there is no need to worry about the internal workings of the source DCS, regardless of the supplier.

The idea that everything should be calculated in the cloud has benefits and drawbacks. An opponent of the idea argued that the plant is actually run based on the values the DCS

gives, so they are the best source of information about the process. Also, such refined calculations as enthalpy and load are of such fundamental importance to the process that their correctness is practically a non-issue; the supplier will surely have ensured their validity, or the process wouldn't even be running. More still, if the process values given by the cloud would deviate even slightly from the ones at the plant, the customers would become suspicious of the whole system, even if the reasons could be explained.

A compromise could be to both gather the DCS calculated values as well as perform the same calculations in the cloud for internal use. This might even provide information about the calculation precision of the DCS, which would also be valuable data for product assessment. The usefulness of such a dual system should however be measured carefully.

5.4 Plant Changes

The latest, final P&IDs are the "As Built" versions, describing the true physical layout of the finished plant. After the delivery phase, however, there might not be information flow backwards to the boiler supplier if the customer decides to alter the process somehow. This could mean that the KPIs defined for the initial plant layout are no longer applicable, and the measurements might not show the values a person managing the Industrial Internet for that plant expects. Some sensors could be replaced and renamed, some valves removed, additional pipelines drawn etc., and as time passes, the history data in the cloud would become more and more skewed. At worst, if this data is used to construct models of boiler performance from multiple data sources, it might actually impact the outcome negatively without the II service provider even knowing.

A crude fix for this would be to demand the customer to inform the tech supplier of any modifications in a process under Industrial Internet info gathering, so the data pipeline can be amended or in the worst case even severed. A more fruitful outcome, however, would be if the customer could be persuaded to share the information about the modifications so the calculation procedures, and even the P&IDs, could be updated and the gathering continued. This may not always be possible.

6. CONCLUSION

This thesis had two primary goals:

- To generalize the engineering procedure for KPI calculation, showcase what KPI calculations are applicable to power boilers, and for what reasons
- To demonstrate, in practice, how the calculation can be implemented and performed, and assess the performance of the system

The first point was handled during the preliminary phase, when data gathering plans were compiled. The data collection and KPI definition processes were performed in parallel, and that was the first step in creating an engineering procedure for data collection. Later, when new plants are being integrated to the VII platform, this procedure will gradually be refined. The data list had to be comprehensive enough to serve both KPI calculation as well as the product engineers, but also remain sensible in scope and storage requirements. The definition of the KPIs influenced the data requirements, and the availability of data dictated which KPIs could be calculated.

KPI calculations applicable to boilers were examined with the help of process experts, who gave valuable input on methodology. A suitable selection of possible KPIs was picked from an expansive list, and their reasoning and calculation methods were showcased.

The second point was handled with the help of Valmet automation experts. After the data sourcing was complete, the data pipeline had to be established, secured and verified. When data was available, the actual system was built ground-up to accommodate the needs of the VII platform. This included more than just writing the script, because a lot of planning had to be made midway: the definition of different auxiliary tables to hold the necessary information in the right format, as well as trying to plan ahead on what sort of structure might be needed in the future. These design choices will affect the flexibility of the calculation system, and only further testing with additional plants will show if a different approach is required.

The script written here, as well as the surrounding computing framework, did perform its intended function. It was able to consistently produce calculation results from the pilot site raw data, which were deemed correct by hand. It was fast in its operation, calculating more than 4 million data points in a matter of 5 minutes, and robust enough not to crash in unexpected conditions. An important aspect in its design was the ease of management, which was handled with a distributed model. In this model different people, who understand the operation of an individual plant, can work with the KPI implementation of that

specific plant. Moreover, it allows the calculation of KPIs based on raw process data from virtually any plant, independent of the internal working or even the supplier of the DCS.

Based on the current performance, this model of automating KPI calculation, as well as the engineering procedure behind it, is adequate for the intended use at Valmet. The immediate next steps would most probably be the addition of more KPIs to the calculation and the inclusion of multiple sites. Besides that, the issues which might impact the usability of the system need to be addressed, most notably the change in design philosophy of processes. To assess the performance of boilers and their individual components, more data is required than the operational point of view provides.

The benefits of the system can be assessed from two angles:

- Its value as a part of the VII product family
- Its value for Valmet in helping make data extraction contracts

KPI calculation is of fundamental importance in every business attempting to track their performance. In this case, the KPIs were very much technical and on-point, describing the status of the physical process somehow. However, from this standpoint, it is easy to expand the KPI set to include more sophisticated, applicable and practical calculation procedures. An immediate example would be to include material costs in the applicable calculations, which would show the financial impacts of different events. Such information is much more tangible than pure process-related information to managers responsible for business-level decision making. KPIs can also be benchmarked against nominal values, which can reveal areas of low performance automatically, as well as provide insight to the reasons for the situation.

Additionally, the KPI calculation set provides an incentive for the customer to allow data extraction from their site and implementation of the data pipeline. Customers might be less willing to allow any data extraction from their sites if it only went for Valmet's internal product quality management and they get no benefit for themselves. The KPI calculation set is both justified in that it is something that could be expected based on the data extracted, as well as the relative simplicity in its implementation. KPI dashboards offer the customer a window to the VII product family and serves as a stepping stone to more advanced products, which can utilize the same data pipeline. In this situation it is much easier to justify using the process data for internal use as well.

The Industrial Internet has seen rapid growth in the recent years due to both technological developments and increase in company awareness. The benefits and opportunities in expanding to the digital market has been recognized in many industrial sectors. Industrial Internet marks a profound shift in the business paradigm of many traditional industries, where focus will increasingly concentrate from device and hardware manufacturing to software management and service providing. Competition is becoming fierce, and only time will tell which businesses can make the next great leap forward.

REFERENCES

[1] A. Tan, How machine learning is applied in industrial IoT, Computer-Weekly.com, 2017, https://www.computerweekly.com/news/450431977/How-machine-learning-is-applied-in-industrial-IoT.

[2] Internet Growth Statistics 1995 to 2017, web page. Available (accessed 16.1.2018): http://www.internetworldstats.com/emarketing.htm.

[3] Lissu Liikenteenseuranta, web page (Finnish). Available (accessed 17.1.2018): http://lissu.tampere.fi/.

[4] Flight Radar 24, web page. Available (accessed 17.1.2018): https://www.flightradar24.com/60.32,25.15/6.

[5] Posti Lähetyksenseuranta, web page (Finnish). Available (accessed 17.1.2018): https://www.posti.fi/henkiloasiakkaat/seuranta/#/.

[6] Smart Home, Market Data Highlights, web page. Available (accessed 17.1.2017): https://www.statista.com/outlook/279/100/smart-home/worldwide#.

[7] Internet of Things (IoT), web page. Available (accessed 17.1.2018): http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[8] What Is the Industrial Internet of Things?, web page. Available (accessed 17.1.2018): https://www.ge.com/digital/blog/everything-you-need-know-about-industrial-internet-things#the-industrial-internet-vs-internet-of-things-3.

[9] Industrial Internet Consortium, web page. Available (accessed 17.1.2018): http://www.iiconsortium.org/index.htm.

[10] Valmet in Brief, web page. Available (accessed 17.1.2018): http://www.valmet.com/globalassets/investors/reports--presentations/other-presentations/2017/valmet-in-brief-2017q3.pdf.

[11] Valmet Energy Production, web page. Available (accessed 17.1.2018): http://www.valmet.com/energyproduction/.

[12] Valmet Competitors, web page. Available (accessed 17.1.2018): http://www.valmet.com/investors/valmet-as-an-investment/competitors/.

[13] ABB Ability, web page. Available (accessed 18.1.2018): http://new.abb.com/abb-ability.

[14] ABB Intenet of Things, Services and People, web page. Available (accessed 18.1.2018): http://new.abb.com/control-systems/features/industrial-IoT-services-people-use-cases.

[15] Andritz Metris, web page. Available (accessed 18.1.2018): https://www.andritz.com/metris-en/home.

[16] Siemens MindSphere, web page. Available (accessed 18.1.2018): https://www.siemens.com/global/en/home/products/software/mindsphere.html.

[17] Siemens Plant Data Services, web page. Available (accessed 1.18.2018): http://www.industry.usa.siemens.com/services/us/en/industry-services/services-glance/plant-data-services/Pages/Digital-Services.aspx.

[18] Valmet Performance Centers, web page. Available (accessed 18.1.2018): http://www.valmet.com/about-us/industrial-internet/valmet-performance-centers/.

[19] World Primary Energy Consumption, web page. Available (accessed 22.5.2018): https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy/primary-energy.html.

[20] R. Raiko, J. Saastamoinen, M. Hupa, I. Kurki-Suonio, Poltto ja palaminen (Finnish), 2nd ed. Teknistieteelliset akatemiat, Helsinki, Finland, 2002, 750 p.

[21] E. Vainio, Fate of Fuel-Bound Nitrogen and Sulfur in Biomass-Fired Indistrial Boilers, Åbo Akademi University, 2014, 95 p.

[22] Anthropogenic emission sources, web page. Available (accessed 25.5.2018): http://elte.prompt.hu/sites/default/files/tananyagok/AtmosphericChemistry/ch02s03.html.

[23] CO2 equivalents, web page. Available (accessed 25.5.2018): https://climatechangeconnection.org/emissions/co2-equivalents/.

[24] U. Jecht, Flue Gas Analysis in Industry, Testo, 14.5.2018, 2008, 149 p., http://www.testo350.com/downloads/Flue_Gas_in_Industry_0981_2773.pdf

[25] K. Rayaprolu, Boilers for Power and Process, 1st ed. CRC Press, Boca Raton, 2009, 745 p.

[26] Solar Steam Power On the Rise, web page. Available (accessed 29.1.2018): https://www.asme.org/engineering-topics/articles/boilers/solar-steam-power-on-the-rise.

[27] Electric Steam Boilers, web page. Available (accessed 29.1.2018): http://www.vaporpower.com/products/electric-boilers/electric-steam-boilers/.

[28] B. Buecker, Steam Generation Thermodynamics 101, Power Engineering, 2008, http://www.power-eng.com/articles/print/volume-112/issue-11/features/steam-generation-thermodynamics-101.html.

[29] Steam Tables Online, web page. Available (accessed 29.1.2018): https://www.steamtablesonline.com/steam97web.aspx.

[30] V. Sathyanathan, Chemical Recovery Boilers in Paper Plants - Part 1, web page. Available (accessed 29.1.2018): http://www.brighthubengineering.com/power-plants/33015-chemical-recovery-boilers-in-paper-plants-part-one/.

[31] Cogeneration / Combined heat and Power (CHP), web page. Available (accessed 29.1.2018): https://www.clarke-energy.com/chp-cogeneration/.

[32] J. Zactruba, How does a Circulating Fluidized Bed Boiler Work?, web page. Available (accessed 29.1.2018): http://www.brighthubengineering.com/power-plants/26547-how-does-a-circulating-fluidized-bed-boiler-work/.

[33] Six Degrees of Fluidization, Powder Bulk, 2016, https://www.powder-bulk.com/wp-content/uploads/pdf/pbe_20161001_0023.pdf.

[34] Fluidised Bed Combustion (FBC), web page. Available (accessed 29.1.2018): https://www.photomemorabilia.co.uk/FBC.html.

[35] HYBEX boilers - using BFB technology, web page. Available (accessed 29.1.2018): http://www.valmet.com/energyproduction/bfb-boilers/.

[36] What are the main characteristics of fluidised bed combustors?, web page. Available (accessed 29.1.2018): http://www.handbook.ifrf.net/handbook/cf.html?id=87.

[37] P. Basu, Circulating Fluidized Bed Boilers: Design, Operation and Maintenance, Springer International Publishing, Switzerland, 2015, 366 p.

[38] Overview of a Valmet CYMIC Boiler, Tampere, Customer Presentation, 2015, 4 p.

[39] D. Pauschert, Study of Equipment Prices in the Power Sector, 122/09, World Bank, Washington, DC, ESMAP technical paper 2009, Available: http://hdl.handle.net/10986/17531.

[40] J. Zactruba, How does a Power Plant Boiler work? - Water and Steam System, web page. Available (accessed 29.1.2018): http://www.brighthubengineering.com/power-plants/23879-how-does-a-power-plant-boiler-work-water-and-steam-system/.

[41] Combustion, web page. Available (accessed 29.1.2018): https://www.grc.nasa.gov/www/k-12/airplane/combst1.html.

[42] J. Zactruba, How does a Power Plant Boiler work? - Combustion System, web page. Available (accessed 29.1.2018): http://www.brighthubengineering.com/power-plants/24054-how-does-a-power-plant-boiler-work-combustion-system/.

[43] Air staging for NOx control, web page. Available (accessed 29.1.2018): http://www.iea-coal.org.uk/site/ieacoal/databases/ccts/air-staging-for-nox-control-over-fire-air-and-two-stage-combustion.

[44] Fuel for Thermal Power Generation, web page. Available (accessed 29.1.2018): http://www.kepco.co.jp/english/energy/fuel/thermal_power/fuel/index.html.

[45] FUELS AND COMBUSTION, Guide Book, National Certificate Examination for Energy Managers and Energy Auditors, 28 p.

[46] R.K. Manfred, Coal-water slurry: A status report, Energy, Vol. 11, Iss. 11, 1986, pp. 1157-1162. https://www.sciencedirect.com/science/article/pii/0360544286900526.

[47] R.K. Rajput, A Textbook of Power System Engineering, Laxmi Publications (P) Ltd., New Delhi, India, 2006, 1085 p.

[48] Backfire Detection, web page. Available (accessed 30.1.2018): https://atexon.com/solutions/backfire-detection/?lang=en.

[49] Exergy Analysis and Efficiency Improvement of a Coal Fired Thermal Power Plant in Queensland, web page. Available (accessed 30.1.2018): https://www.intechopen.com/books/thermal-power-plants-advanced-applications/exergy-analysis-and-efficiency-improvement-of-a-coal-fired-thermal-power-plant-in-queensland#F1.

[50] Kilpilahti Combined Heat and Power Project, web page. Available (accessed 30.1.2018): https://www.power-technology.com/projects/kilpilahti-combined-heat-power-project/.

[51] What is Distributed Control System (DCS)?, web page. Available (accessed 30.1.2018): https://www.electricaltechnology.org/2016/08/distributed-control-system-dcs.html.

[52] Valmet DNA automation system, web page. Available (accessed 30.1.2018): http://www.valmet.com/automation-solutions/valmet-dna-dcs/valmet-dna-automation-system/.

[53] Valmet DCS overview, Customer Specification, 2014, 6 p.

[54] Instrumentation & Control. Process Control Fundamentals, pacontrol.com, Unpublished Document.

[55] From Gauges to Transmitters, Flow Control, Magazine Article, 2011, https://www.flowcontrolnetwork.com/from-gauges-to-transmitters/.

[56] J. Collin, A. Saarelainen, Teollinen internet, Talentum, Helsinki, Finland, 2016, 333 p.

[57] A. Nopanen, Utilizing the Industrial Internet for Power Plants, Tampere University of Technology, 2017, Available: http://dspace.cc.tut.fi/dpub/handle/123456789/25242.

[58] Cloud Computing, web page. Available (accessed 30.1.2018): http://searchcloudcomputing.techtarget.com/definition/cloud-computing.

[59] What is the Cloud? – Definition, web page. Available (accessed 30.1.2018): https://www.sdxcentral.com/cloud/definitions/what-is-cloud/.

[60] Dropbox, web page. Available (accessed 30.1.2018): https://www.dropbox.com/?landing=dbv2.

[61] What can you do with Gmail?, web page. Available (accessed 30.1.2018): https://gsuite.google.com/learning-center/products/gmail/get-started/.

[62] Office 365 Service Descriptions, web page. Available (accessed 30.1.2018): https://technet.microsoft.com/fi-fi/library/office-365-service-descriptions.aspx.

[63] When to use SaaS, PaaS, and IaaS, web page. Available (accessed 30.1.2018): https://www.computenext.com/blog/when-to-use-saas-paas-and-iaas/.

[64] What is Cloud Computing? web page. Available (accessed 30.1.2018): https://aws.amazon.com/what-is-cloud-computing/.

[65] What is Cloud Computing? web page. Available (accessed 30.1.2018): https://www.ibm.com/cloud/learn/what-is-cloud-computing.

[66] Landsnet KKS Handbook, Unpublished material, 2016, https://www.landsnet.is/library/Skjol/Flutningskerfid/Um-flutning-skerfid/KKS/KKS%20handbook%20English%20-%20november%202016.pdf.

[67] KKS: Kraftwerk-Kennzeichen-System, Identification System For Power Plants, Siemens, Unpublished material.

[68] KKS - Identification System for Power Plants, web page. Available (accessed 30.1.2018): http://www.kronebach.com/kks/e/index-e.html.

[69] How a Process Control Loop Works in Automatic Control Systems, web page. Available (accessed 30.1.2018): http://www.instrumentationtoolbox.com/2012/01/how-process-control-loop-works-in_24.html.

[70] Valmet DNA Engineering Function Block CAD, web page. Available (accessed 30.1.2018): http://www.valmet.com/automation-solutions/valmet-dna-dcs/valmet-dna-automation-system/engineering-and-maintenance-tools/valmet-dna-engineering-function-block-cad/.

[71] Key Performance Indicators, web page. Available (accessed 31.1.2018): http://www.synsysinc.com/key-performance-indicators.asp.

[72] C. Schmidt, W. Li, S. Thiede, B. Kornfeld, S. Kara, C. Herrmann, Implementing Key Performance Indicators for Energy Efficiency in Manufacturing, Procedia CIRP, Vol. 57, 2016, pp. 758-763. https://www.sciencedirect.com/science/article/pii/S2212827116312914.

[73] C. Lindberg, S. Tan, J. Yan, F. Starfelt, Key Performance Indicators Improve Industrial Performance, Energy Procedia, Vol. 75, 2015, pp. 1785-1790. http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-29328.

[74] Pilot Site P&I Diagram, Technical Specification, 2015, 64 p.

[75] Pilot Site Control Diagram, Technical Specification, 2015, 187 p.

[76] Pilot Site Interlocking Diagram, Technical Specification, 2015, 188 p.

[77] Steam Boiler - Maximum Continuous Rating, web page. Available (accessed 14.5.2018): http://steamofboiler.blogspot.fi/2011/08/maximum-continuous-rating-mcr.html.

[78] Boiler Shutdown, web page. Available (accessed 14.5.2018): http://www.banksengineering.com/shutdown.htm.

[79] Control And Protection - Master Fuel Trip, web page. Available (accessed 14.5.2018): http://wbpdclewf.org.in/wp-content/uploads/2016/09/Control___Protection.pdf.

[80] E. Vakkilainen, Steam Generation from Biomass: Construction and Design of Large Boilers, Elsevier Inc., United Kingdom, 2016, 322 p.

[81] T. Mathaeus Own Power Consumption, web page. Available (accessed 14.5.2018): https://www.abb-conversations.com/2013/08/optimizing-power-plant-performance-for-energy-effiency/.

[82] Industrial Emission Regulations in Finland, web page. Available (accessed 14.5.2018): http://www.ym.fi/en-US/The_environment/Climate_and_air/Protection_of_the_climate_and_the_ozone_layer/International_cooperation_and_EU_affairs.

[83] Boiler System Energy Losses, web page. Available (accessed 14.5.2018): https://www.nrcan.gc.ca/energy/efficiency/industry/technical-info/tools/boilers/5431.

[84] Vedenkäsittely - Water Treatment (Finnish), Lappeenranta University of Technology, 14.5.2018, 2013, 92 p.

[85] Flue Gas Recirculation for NOx Reduction, 14.5.2018, 4 p.

[86] Common Causes of Vibration in Centrifugal Fans, web page. Available (accessed 14.5.2018): http://www.cecvp.com/latestnews/common-causes-of-vibration-in-centrifugal-fans/.

[87] Aginity Workbench, web page. Available (accessed 29.5.2018): https://www.brightgrove.com/projects/aginity-workbench/.

[88] Amazon Glacier, web page. Available (accessed 6.6.2018): https://aws.amazon.com/glacier/.

[89] G. Prentice Temperature Measurement Accuracy Guidelines, web page. Available (accessed 13.8.2018): https://www.flowcontrolnetwork.com/temperature-measurement-accuracy-guidelines/.

[90] Accuracy, precision & resolution, web page. Available (accessed 17.8.2018): https://meettechniek.info/measurement/accuracy.html.

[91] T. O'Haver Signals and Noise, web page. Available (accessed 17.8.2018): https://terpconnect.umd.edu/~toh/spectrum/SignalsAndNoise.html.

[92] G. Helms, P. Dechent, Increased SNR and Reduced Distortions by Averaging Multiple Gradient Echo Signals in 3D
FLASH Imaging of the Human Brain at 3T, JOURNAL OF MAGNETIC RESONANCE IMAGING, Vol. 29, 2009, pp. 198–20. https://onlinelibrary.wiley.com/doi/pdf/10.1002/jmri.21629.