Infrastructure for Collaborating Data-Researchers in a Smart Grid Pilot

W. Labeeuw, *Graduate Student Member, IEEE*, S. Claessens, K. Mets, C. Develder*, Member, IEEE*,
G. Deconinck*, Senior Member, IEEE*

***Abstract*—A large amount of stakeholders are often involved in Smart Grid projects. Each partner has its own way of storing, representing and accessing its data. An integrated data storage and a joint online analytical mining infrastructure is needed to limit the amount of duplicated work and to raise the overall security of the system. The proposed infrastructure is composed of standard application software and an in-house developed data analysis tool that allows researchers to add and share their own functionality without compromising security.**

***Index Terms*—Smart Grid, Software architecture, Data analysis, Data storage systems, Data warehouses**

I. Introduction

A lot of data has to be processed in smart grid projects. Data for analysis is provided by different partners with different approaches towards storing and accessing data. The need for an integrated tool emerges. An infrastructure for conducting analytical mining by different researchers in different universities and institutes is presented in this paper. The focus of the infrastructure is functionality for the researchers and security of the data to meet the concerns of the data providing partners while respecting the knowledge discovery and data mining process. The implementation of the infrastructure is done by using application software and an in-house developed software tool. A central software version control system hosts the in-house developed software tool.

Within the 'Linear'-project [1], different parties have provided data to perform initial analyses upon. The data consists of metering and questionnaire information. Electricity consumption as well as generation at residential level is metered. A subset of the metered households are questioned at home. Responses to the questionnaire are coupled with the metered electricity consumption to assess demographic properties.

The data providing parties are concerned about the security of the data. Especially confidentiality needs to be tackled. Only authorized researchers have access to the data or parts of the data. A central data storage system is the easiest way to secure data [2]. Researchers, on the other hand, are interested in availability and the functionality of the data analysis tool. The tool should have a limited impact on their desktop, but the availability of the data should be high. Most of the data transformation will need to be done at the server side and only the results are stored locally. Data transformation and retrieval from the central storage should be automated and their preferred machine learning tool should be able to read the retrieved data.

II. Requirements

The requirements of the infrastructure are, as mentioned above, twofold. The data providing parties have a strong focus on security: only research institutes with a non-disclosure agreement with the data providing party are allowed to read the data. Researchers, on the other hand, want functionality: the infrastructure and the corresponding software tool need to respect the knowledge discovery and data mining process. Automation and processing data transformations at the server side are regarded as an asset.

A. Security

Security patterns are a mix of design patterns with focus on security and best practices with respect to the architecture and the system [2]. Although several security patterns exist, not all of them are well supported in the pattern landscape [3]. Literature describes four kinds of security properties that are commonly accepted: confidentiality, integrity, availability and accountability [4], [5]. Confidentiality ensures that only authorised people or software are able to view the data or certain parts of the application. By enforcing integrity, data can only be adjusted by authorised people or software and authorisation is required to execute application calls. Availability makes sure that the system is accessible and responsive at all times. Accountability keeps track of who is responsible for which action and can be used for non-repudiation and auditing.

The main requirements for the data providing partners are related to data confidentiality. Researchers are interested in availability and integrity at the server side. They want to be able to continue their work if the server goes down and the data has to remain sound to make sure that the results of the analyses are reliable. Further on, someone should be held accountable if something goes wrong with confidentiality and integrity.

This work is supported by the Ministry of Science (Minister I. Lieten) via the project Linear organized by the Institute for Science and Technology (IWT).

Wouter Labeeuw and Geert Deconinck are with the Department of Electrical Engineering ESAT-ELECTA, KU Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium (e-mail: wouter.labeeuw@esat.kuleuven.be and geert.deconinck@esat.kuleuven.be).

K. Mets and C. Develder are with the Department of Information Technology Broadband Communication Networks (IBCN), Ghent University - IBBT, Gaston Crommenlaan 8 (Bus 201), B-9050 Gent, Belgium (email: kevin.mets@intec.ugent.be and chris.develder@intec.ugent.be).

S. Claessens is with VITO, Boeretang 200, B-2400 Mol, Belgium (email: sven.claessens@vito.be).

B. Functional

The infrastructure has to meet the requirements for the knowledge discovery and data mining (KDD) and support the online analytical mining (OLAM) processes. Online analytical mining is online analytical processing (OLAP) with the inclusion of data mining.

Within online analytical processing of data, two subtypes are defined: relational and multidimensional OLAP [6], [7]. Relational processing works with data stored in relational databases. SQL-queries are used to process the data. Multidimensional processing tools store data in special data structures such as arrays and matrices. They use matrix operations to get the data into the right form. Most of the processing needs to be done in the relational database using SQL because of the flexibility of the SQL-language. Special operations that are hard to describe in SQL should be done by special routines on the multidimensional data store.

The knowledge discovery and data mining process has five different steps [8]: data selection, data preprocessing, data transformation, data mining and interpretation. Data selection is done on beforehand by consensus. The preprocessing of the data is done before the data is placed in the database. OLAP ensures the data transformation. Data transformation results are stored locally. The use of a local computer makes it easier for a researcher to use in-house developed or specific data mining tools. In general, only subsets of the data are required by the researcher. By keeping data at a central point and requesting specific data, data storage at the researchers part is reduced. However, not having all the data locally forces the researcher to query each time some required data is not yet available locally.

Automation of data retrieval makes it easier to query for data. Researchers should be able to select data based on the relations of the different questionnaire responses and metering data in the database. Data needs be transformed into the right aggregated form and time dimensions when it reaches the researcher. It is preferred that the communication between the central server and the researcher is limited. This requires data transformations to be done at the server side by the central SQL-server, i.e. relational online analytical processing. Transformations that can not be executed by the SQL-server because of the complexity of query generation, are done at the desktop of the researcher. First, SQL-queries are sent to the server to pre-transform the data. The received pre-transformed data is then converted locally into the correct form by matrix transformations, which is multidimensional online analytical processing.

Researchers should be able to make adjustments to the data analysis tool. The source code of the software tool has to be shared by using a software version control system. Sharing the source code increases the productivity of the researchers: already implemented data retrieval or data transformation scripts can be reused and adapted by other researchers.

III. Overview

An overview of the infrastructure can be found in Figure 1. Three different stages are defined: data cleaning and preprocessing, data storage and data transformation, analysis and interpretation according to the steps of the knowledge discovery and data mining process. The full data set is stored at a central server in an SQL-repository. An in-home developed data analysis tool is marked in grey. The tool includes cleaning, warehousing (SQL-query generator), scripting and processing. A version control system that keeps track of the adjustments in the software tool is also hosted centrally. Researchers have a version control system client to keep the software tool locally up to date and to be able to commit their own adjustments.

A. Data analysis tool

In Figure 1, the software tool is indicated by the grey frame that surrounds its functionality. The different steps of the knowledge discovery and data mining process determine the functionality of the developed software tool.

At the local cleaning and preprocessing side, data selection, data cleaning and data preprocessing is executed. Data selection is done on beforehand by consensus in meetings between researchers and data providing parties about what data should be researched. The cleaning functionality of the software tool performs cleaning and preprocessing. The version control system is able to keep code secret based on authentication and authorisation. By placing data provider specific cleaning functions in a different directory and by limiting authorisation on the directories, references to specific data structures of the data providing party are kept confidential.

Data transformation and interpretation are controlled at the local transformation, analysis and interpretation side. The warehousing functionality of the software tool generates SQL-queries to process relations in the data and to aggregate data based on date and time criteria. The results of the warehousing are stored in the local data store, preserving the integrity of the central data. Not all of the required data transformations can be done by SQL queries. Processing functionality of the software tool implements some additional transformations. Also, visualisation is implemented. The processing functionality allows for automated plotting of processed and transformed data. The resulting plots are again stored at the local data store.

B. Security

The central storage server hosts the SQL-server. A firewall, indicated in dark grey on Figure 1, limits the range of IP-addresses that can connect to the central storage server. Only the IP-addresses of the researchers, or the research institutes if IP-addresses are dynamical, are allowed to connect to the server. To secure the connection between the SQL-server and the different clients, the SQL-traffic is tunnelled using secure shell (SSH). It is not possible to eavesdrop on or to tamper the SQL-messages in an SSH-tunnel. By storing the RSA authorisation keys for the SSH connection to the server locally, the researcher will be informed about an invalid key if someone is trying to spoof the IP-address of the server. Another advantage is that the SQL-server only needs allow connections from $localhost$. The connection with the version control system is secured by Hypertext Transfer Protocol

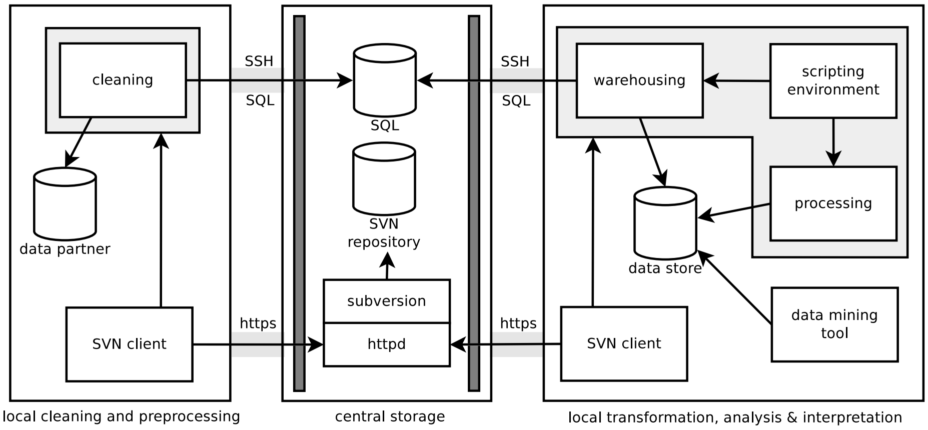

Fig. 1. Data analysis infrastructure

Secure (HTTPS), which is HTTP traffic with Transport Layer Security (TLS). Again, eavesdropping and tampering is made almost impossible. A certification authority (CA) ensures that spoofing is impossible.

IV. Security

The most important requirement regarding security is confidentiality. The data providing parties want to limit the number of people that have access to their data. Only research institutes with a non disclosure agreement with the data providing party is able to view the data. Next, data integrity at the central storage has to be met. If the data got changed before analysis, wrong conclusions will be drawn. Availability is important for the researchers. If the central server goes down, data analysis can still be performed on the locally stored data. And if confidentiality and integrity are violated, it should be possible to trace back where the violation occurred.

Confidentiality is applicable to the data and the application. Transmission confidentiality makes eavesdropping impossible. Communication is made confidential by working with secure pipes and secure access layers between the central storage and the local workstations. Storage confidentiality at the central server can be done by encrypting the data storage, but the double authentication (SSH and SQL) and authorisation (firewall) should keep unwanted guests out. Confidentiality of the local data store is the responsibility of the researcher. Application confidentiality is hard to implement if each client has access to the data analysis tool. However, to make sure that the data remains confidential, researchers and data providers are only granted a limited access on the different tables. An SQL-error will be generated if someone is trying to access data without authorisation to do so.

The integrity of the data and the applications are tackled as well. Transmission integrity is also dealt with by secure pipes and secure access layers, making tampering impossible. Storage integrity at the central storage is assured because researchers only have read access in the database and the transformed data used for mining is stored locally. Data providers only have access to their data. Application integrity can be coped with by using an input guard that checks all SQL queries before passing them to the SQL-server. An SQL-server has safety measures like authentication and authorisation on its own and generates errors if erroneous queries are send to the server. Further on, a query log keeps track of who is executing which queries. It will be easy to point out which researcher executed malicious queries.

A firewall makes sure that only connections from authorised IP-addresses can be established and that only connections to certain ports on the server are allowed. The availability is raised by storing data transformation results locally. Accountability is tackled by logging SQL-queries and SSH login attempts.

A. Storing data at the client

In most cases, it is preferred to store data at the server side for security reasons [2]. However, in the proposed infrastructure, client storage has advantages. Data providing parties are able to chose which data is made available to the central server. Not all metered data is placed in the central server. Confidentiality of the non-upload data is as a consequence high. The data at the central storage remains sound because researchers are only limited to manipulating data locally. Also, by storing the transformed data locally in comma separated value files, it is possible to use most modern data mining tools. Local storage makes sure that the already requested data stays available if the server goes down.

The software tool needs to stay sound as well. Researchers and data providing parties are able to make adjustments to the data analysis tool locally. After tests, changes can be committed to the central storage. If something goes wrong with the data analysis tool, a previous version will replace the non-functioning software. The data analysis tool remains operational in this way. It is also possible for the data providing parties to keep the specific functionality to pre-process their own data locally, without committing it to the central server.

B. Firewall

A firewall operates at system level and is used to restrict access to certain hosts at network level and to control incoming and outgoing network connections [9]. The firewall is configured in such a way that only connections from IP-addresses of researchers and data providers are allowed. In cases where IP-addresses are dynamic, a range of addresses is used in the firewall, which is a practical, but less secure solution. Only SSH- and HTTPS-connections are allowed, since these are the only services that should be accessible from outside the server. However, a firewall does not protect against other SSH tunnelled connections. Firewalls are also unable to prevent misuse of SQL and subversion. An input guard is able to fix this problem, but because of the limited amount of users and the slow query log, it was not implemented. A slow query log tracks executed SQL-queries and the responding user that executed the queries, making repudiation impossible.

C. Secure access layer

Secure access layer is a non-application security solution [10]. Existing security infrastructure is used to make insecure applications secure, without adapting the application. The secure shell (SSH) connection layer is used to tunnel SQL-connections. It is impossible to eavesdrop on the tunnelled connection and the tunnelled connection can not be tampered [11]. The use of SSH requires each researcher or research institute to have an account on the host. By using a secure access layer (SSH-tunnel), a standard SQL-connection can be created between the application at the side of the researcher or the data providing partner and the SQL-server. The SQL-server only needs to listen to the *localhost* because all connections have to be tunnelled. Connections that are not tunnelled are not allowed, which increases security.

The disadvantage of secure shell is that users get an account on the server. They are able to execute applications on the server and they might try to get root access. The root account needs to be disabled, so that nobody outside of the project tries to guess the root password. This means that another account has to get the possibility to become administrator (super user).

Most SSH-clients have a repository for the RSA-keys that are used to establish the connection. When the RSA-keys are stored locally, researchers and data providing parties will be notified that the key is invalid if they try to connect to a spoofed server, which is a way of detecting man-in-the-middle attacks.

D. Secure pipe

A secure pipe tackles the same security issues as a secure access layer: transmission confidentiality and integrity. However, secure pipes are, in contrast to secure access layers, a form of application security [12]. Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), provide secure transmission [13].

Apache Subversion is selected as version control system because it is a centralized rather than distributed version control system. With a central system, there is only one repository, which is easier to track. Subversion is able to work together with the Apache web server, by using Web Distributed Authoring and Versioning (WebDAV). The Apache web server also supports Transport Layer Security. The combination of subversion with WebDAV and TLS/SSL results into a version control system with secure communication. Man-in-the-middle attacks are coped with by a public key certificate of the central server, issued by a certification authority. The subversion client uses the public key to encrypt the data. Only the owner of the private key, i.e. the central storage server, is able to decrypt it.

E. Full view with errors

Full view with errors is a security pattern that describes a system where users are allowed to access all software operations, but the system generates error messages when the user performs illegal actions [10]. The pattern is used to disallow users to execute operations where they do not have permissions for. A better way to handle the illegal operations problem is working with a limited view. A limited view gives an overview of the operations that the user is allowed to do [10]. This is impossible to implement, since every researcher has access to the same data analysis tool. Although researchers are informed about the data they can access, full view with errors might frustrate some users because they can view functionality that they can not use. Full view with errors is however easier to maintain.

Each researcher has its own account for the SQL-server. The permissions for viewing the data can vary amongst the different researchers. A researcher is granted its own limited view on the data, which ensures confidentiality. The same goes for the data providing partners: they have their own account and limited access to the database. Due to the fact that some competing companies had access to the same tables, it was opted to appoint someone with write access to the database. A better way to let the data providing partners upload their data, is by the implementation of web services. The next step in the project is to meter and sub-meter households and to control appliances. Web services will be implemented when data has to be uploaded frequently.

V. Data analysis tool

Researchers demand that the data analysis tool is able to conduct online analytical mining, a combination of online analytical processing and data mining. The different steps of the knowledge discovery and data mining paradigm are the basis for the tool. The five steps are data selection, data preprocessing, data transformation, data mining and interpretation. Data mining itself is done by specialised tools and is beyond the scope of the data analysis tool. Online analytical processing relates to relational and multidimensional data. Relations are used to select the relevant data sets for data transformation. The result of the data transformation is multidimensional data that can be read by the data mining tool.

A. Cleaning

Although a large questionnaire is included, most data is metering information of gas consumption or electricity consumption and production. Metered electricity consumption data of over 1500 residential customers, metered gas consumption of over 800 residential customers, sub-metering data of over

80 residential customers and metered data of PV and CHP-installations are amongst the data for the database [14].

Cleaning and pre-processing happens locally. Consensus about the relevant data that needs to be placed into the database determines data selection. Generic implemented functionality to load the data into the database and a fixed database scheme makes sure that the data is delivered uniformly by the different partners. Data providing parties implement the specific functionality to pre-process their own data.

The cleaning module in the data analysis tool is designed to have generic classes to parse data files. At the top, generic classes that can be used across different projects are defined. These classes include routines to automatically generate queries to place data into a database. Queries to place data into the database of the project are the same amongst the different data providing parties and are based on the fixed structure in the SQL-database. Classes that implement the project specific queries, inherit from these generic classes. Missing data, metering information in a different time step and data in a different unit are handled by generic cleaning routines. Data providing parties implement specific classes, tailor made on their specific representation of the data, e.g. a special representation of timestamps, by inheriting from the project classes and using the generic cleaning routines.

The purpose of preprocessing, which is part of the cleaning module in the software tool, is getting the units of the data in the same domain. All data should be in the same energy or in power form. Time intervals of the data have to be the same, e.g. fifteen minute time intervals. Daylight savings time has to be applied in the same way, which requires interpolations and aggregations in certain data sets.

The implementation of the cleaning functionality of the data analysis tool is done in Python. A class designed around the *csv*-library is used to read in the data. For each type of data, a class is implemented to load the data into the database. The idea was that the data provider would define a class that inherits from this database inserting class and implement specific routines for their data. However, this responsibility was assigned to a dedicated database manager. Only the database manager is allowed to write data into the database, which ensures the storage integrity of the data and improves accountability.

B. Warehousing

A data warehouse is, in general, a database which is used for analysis and reporting. In contrast to most warehousing solutions, warehousing data is stored locally in the described infrastructure. Warehousing in this paper refers to a module that implements the relational online processing. The module generates queries to get aggregated data from the central database and stores this information locally at the system of the researcher.

The warehousing module generates SQL-queries to aggregate data. SQL (Structured Query Language) supports multiple aggregation functions, such as minimum, average, standard deviation, variance, sum etc. SQL databases also have a standardized way of storing date and time information and a lot of functions to manipulate that information. Second, minute, hour, day, day of the week, month, etc. information is easy to extract by using SQL. Data can be aggregated over time frames by combining the ‘group by’-statement with date and time intervals. For example, the load curve of the average day of each month in 2008 can be calculated by using the average function, by grouping by minute, hour, and month and by requiring that only data from 2008 is selected, as shown in Figure 2. Functionality like this is ideal to create different kinds of load curves and to analyse behaviour. The date and time data transformations themselves are hard to implement in software, the use of an SQL-queries makes it very convenient.

Query:

```
SELECT
  AVG(power) as p,
  month(ts) as mont, hour(ts) as hou,
  minute(ts) as minut
FROM
  metering
WHERE
  ts BETWEEN %s AND %s
GROUP BY
  mont, hou, minut
ORDER BY
  mont, hou, minut
```

with arguments (Python, the datetime library):

- `datetime(2008,1,1)`
- `datetime(2009,1,1) - timedelta(seconds=1)`

Fig. 2. Automatically generated SQL-query to retrieve the load curve, detailed up to minutes, of the average days of each month for all metered profiles in 2008

The version control system makes sure that all researchers have access to the warehousing module. Each researcher can implement its own warehousing queries into the module. Researchers tend to focus mainly on one domain, e.g. gas or electricity. By pushing the query generating functionality to the parent classes once the functionality has been tested, the functionality becomes available for researchers with focus on other domains, which ensures a rapid development of the data warehousing functionality.

Although a large number of NoSQL-solutions provide SQL-like support (e.g. Hive [15], Pig Latin [16] and HadoopDB [17]), the magnitude of the amount of data is not large enough to justify the use of NoSQL technologies for this infrastructure.

C. Processing

The warehousing module of the data analysis tool is limited to relational processing and the aggregation of data in the time domain. Multidimensional online analytical processing is done by the processing module. The multidimensional processing functionality fills up the gaps of the warehousing: extra date/time transformations, multidimensional algebraic transformations, statistics and plotting functionality. The results from the data mining tool can be processed as well, making it possible to iterate over warehousing, mining and interpretation. Just as for the warehousing module, researchers are encouraged to implement their own algorithms and share them by committing the new functionality to the software repository.

SQL-queries become very complex if a distinction between weekdays and weekends has to be made. The process module is able to read in aggregated data from the warehousing of different weekdays and to convert it into a general weekday and weekend day while keeping the amount of detail available from the warehousing. For example, the average week of the year with detail up to minutes can be converted into the average weekday and weekend day with the same detail. This conversion makes it easier to plot data, given that weekdays are in general similar to each other.

Multidimensional data transformations are easier to perform in an mathematical environment than an SQL-environment. SciPy[1], and its basis NumPy, provide convenient and fast multidimensional array manipulations. SciPy and NumPy were amongst the main reasons to chose Python. Principal component analysis and multidimensional array projections are amongst the most important multidimensional data transformation functions in the processing functionality.

Statistical information is useful to get additional insights of the data. Different kinds of (cumulative) probability density functions and curve fitting functions are implemented to allow for statistical analyses.

The visualisation of the data transformation and the data mining results is done by the plotting functionality in the processing module. Support for the different date/time transformation results is included. An example of the automation of plotting is shown in Figure 3. The $TimeSequencePlot$ class reads the load curve data (measurements) and makes a a plot of the load curve for each selected household, gives the generated plot the name of the household id and places the plots in the $images$ directory. The plot contains curves for each weekday.

D. Scripting

Building all the automation functionality into a graphical user interface would be very time consuming and complex. The functions of data transformation and retrieval depend on the new insights researchers get by analysing the data. A scripting environment is more flexible than a graphical user interface, but requires more effort from the researchers as a trade off. However, it is often easier to change a line of code to get a different data transformation, than to rerun through a graphical user interface. Also, scripts are exchangeable, while user input is not.

Figure 3 shows a generic example of an automated data retrieval script in the developed software package. The script selects the household that meet the first and the second option of question 1.1 in the question series 1. The load curve of the electricity consumption on the average day of the month for each month of 2008 is calculated for these households and saved into the measurements file. The detail of the load curve goes up to minute level. A plot of the load curve of the electricity consumption of each selected household is generated in the folder $images$.

```
hQuest = QuestionRange1()
hQuest.select = ["H_ID"]
hQuest.question1_1 = [1,2]
ids = hQuest.execute()

duration = {"from": datetime(2008,1,1),
  "to": datetime(2009,1,1) - timedelta(seconds=1)}
intervals = ["quarter","weekday","hour","minute"]

eMeasure = Measurements()
eMeasure.selection = ["E"]
eMeasure.duration = duration
eMeasure.aggregator = "avg"
eMeasure.intervals = intervals
eMeasure.h_ids = ids
eMeasure.getdata("output/measurements.csv")

readin = CSVReader("output/measurements.csv")
p = TimeSequencePlot(readin)
p.format = "png"
p.outputdir = "images/"
p.output = map(str, ids)
p.duration = duration
p.newgraphon = "weekday"
p.intervals = intervals
p.execute()
```

Fig. 3. Script example

VI. Related work

Rusitschka et al. presented a smart grid data cloud to manage real time streams [18]. The focus of their work is on different industrial parties in the smart grid domain. As in the infrastructure in this paper, a distinction between parties that are allowed to put data in and the ones that can only get specific elements out of the database is made. However, the interface is provided by APIs instead of a query language (SQL) as presented in this paper. The API interface is based on REST (representational state transfer). A PUT request pushes data into the infrastructure, while a GET request is able to pull data from the server. SQL is, in contrast to REST APIs or portals, a declarative language that gives a lot of flexibility. Their solution is better in an industrial environment where predefined ways of retrieving data are preferred. The approach presented in this paper suits researchers better.

A system infrastructure to analyse data in power grid companies can be found in [19]. Some data mining algorithms are implemented to predict wind power output. The infrastructure has similarities with the infrastructure presented in this paper, although the audience for their system is different. The users of the system are not able to define their own queries, only a portal is available.

Research collaboration is not limited to data. Liu et al. describe a way to let dispatchers, security engineers, planners, etc. collaborate on simulation models [20].

An in depth description of data cleaning of locally and globally corrupted data can be found in [21]. Locally corrupted data is data that deviates from the local patterns of time series, while globally corrupted data deviates from the global trends. Chen et al. use Splines and Kernel Smoothing to clean up the data. Local and global cleaning in this work mainly consists of removing or marking invalid data. Local cleaning is done in the cleaning and preprocessing step, global cleaning is done in the processing step.

[1] SciPy is open-source scientific software for Python, more information available at http://www.scipy.org/

VII. Conclusions

A lot of data mining is happening in the power system and smart grid context [22]–[24]. Most online analytical mining solutions presented in literature target industrial companies and do not allow for active development and cooperation amongst the users of the system [6], [18], [19]. Researchers have different demands with respect to data analysis than people in industry. The paper shows that a flexible infrastructure for collaborating researchers is able to meet the security requirements of industry.

The four types of security patterns (confidentiality, integrity, availability and accountability) are the basis to design the secure infrastructure. The use of secure access layers and a secure pipes enforces confidentiality and integrity of the data connections, making eavesdropping and message tampering impossible. Researchers only get read access on the central database, which makes it impossible for them to change data at the central storage. The authentication systems, SSH- and SQL-authentication, restrict access. Researchers have a full view with errors, which means all the possibilities of the software can be viewed by the researcher, but the software generates errors when the user is trying to access a function without the proper authorisation. The availability of the system is raised by storing the data transformation results locally. However, local storage has some confidentiality risks. A non disclosure agreement of the research institutes with the data providing parties and logging enforces accountability and makes sure that reseachers protect their data locally.

Next to secure, the data analysis infrastructure needs to be functional. The knowledge discovery and data mining process is the basis for the functionality of the infrastructure. The infrastructure has the different steps in the KDD process (data selection, data preprocessing, data transformation, data mining and data interpretation). Data selection is done on beforehand by consensus in meetings. However, it is still possible to select subsets of the data in the data transformation step. Cleaning and preprocessing is implemented in the cleaning module of the data analysis tool. Most of the data transformation is done by the warehousing module. Warehousing, in the context of the data analysis tool, refers to the relational online analytical processing (ROLAP) of the data. The result of the ROLAP process is multidimensional data, which is stored into the local data store. The multidimensional data can be processed or directly used in the data mining tool. Data mining is not implemented in the tool, because most researchers have their own preferred tool and the implementation of different algorithms is very time consuming. Data transformations that are not possible in SQL, are done by the processing module. The processing module implements multidimensional online analytical processing (MOLAP) and includes functions as specific data/time transformations, multidimensional algebraic transformations and statistics. The visualisation, needed for the interpretation of the results, is done by the data mining tool of the researcher or by the processing module.

References

[1] E. Peeters, C. Develder, J. Das, J. Driesen, and R. Belmans, "Towards a breakthrough of smart grids in Flanders," in *i-SUP conference*, Bruges, Belgium, April 2010.

[2] D. Kienzle, M. Elder, D. Tyree, and J. Edwards-Hewitt, "Security patterns repository, version 1.0," 2006.

[3] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, "An analysis of the security patterns landscape," in *Software Engineering for Secure Systems, 2007. SESS '07: ICSE Workshops 2007. Third International Workshop on*, May 2007.

[4] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, no. 5, pp. 35–47, 2008.

[5] A. Yautsiukhin, R. Scandariato, T. Heyman, F. Massacci, and W. Joosen, "Towards a quantitative assessment of security in software architectures," in *Nordic Workshop on Secure IT Systems (NordSec)*, October 2008.

[6] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *SIGMOD Rec.*, vol. 26, no. 1, pp. 65–74, Mar. 1997.

[7] H. Hasan and P. Hyland, "Using OLAP and multidimensional data for decision making," *IT Professional*, vol. 3, no. 5, pp. 44 –50, Sep/Oct 2001.

[8] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, Nov. 1996.

[9] M. Schumacher, "Firewall patterns," in *The 8th European Conference on Pattern Languages of Programs (EuroPLoP 2003)*, Irsee, Germany, June 2003.

[10] J. W. Yoder and J. Barcalow, "Architectural patterns for enabling application security," in *Fourth Conference on Pattern Languages of Programs (PLoP 1997)*, Monticello, Illinois, Sept. 1997.

[11] T. Ylonen and C. Lonvick, "The secure shell (SSH) transport layer protocol," *Request for Comments 4254*, 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4253.txt

[12] C. Steel and R. a. Nagappan, *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management*. Prentice Hall PTR, 2005.

[13] D. Wagner and B. Schneier, "Analysis of the ssl 3.0 protocol," in *Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, ser. WOEC'96. Berkeley, CA, USA: USENIX Association, 1996, pp. 29–40.

[14] W. Labeeuw and G. Deconinck, "Customer sampling in a smart grid project," in *Proceedings of the IEEE PES general meeting*, 2012, accepted, to be published.

[15] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: a warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1626–1629, Aug. 2009.

[16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1099–1110.

[17] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 922–933, Aug. 2009.

[18] S. Rusitschka, K. Eger, and C. Gerdes, "Smart grid data cloud: A model for utilizing cloud computing in the smart grid domain," in *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2010, pp. 483 –488.

[19] A. Bara, I. Lungu, M. Velicanu, and S. Oprea, "Intelligent systems for predicting and analyzing data in power grid companies," in *Information Society (i-Society), 2010 International Conference on*, june 2010, pp. 266 –271.

[20] J. Liu, X. Li, D. Liu, H. Liu, and P. Mao, "Study on data management of fundamental model in control center for smart grid operation," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 573 –579, dec. 2011.

[21] J. Chen, W. Li, A. Lau, J. Cao, and K. Wang, "Automated load curve data cleansing in power systems," *Smart Grid, IEEE Transactions on*, vol. 1, no. 2, pp. 213 –221, sept. 2010.

[22] M. Sforna, "Data mining in a power company customer database," *Electric Power Systems Research*, vol. 55, no. 3, pp. 201 – 209, 2000.

[23] L. B. Romdhane, N. Fadhel, and B. Ayeb, "An efficient approach for building customer profiles from business data," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1573–1585, 2010.

[24] G. Chicco, R. Napoli, and F. Piglione, "Comparisons among clustering techniques for electricity customer classification," *Power Sytems, IEEE Transactions on*, vol. 21, no. 2, pp. 933 – 940, 2006.

VIII. Biographies

Wouter Labeeuw received the M.Eng. degree in electronics-ict from Provinciale Industriële Hogeschool, Kortrijk, Belgium, in 2007 and the M.Sc. degree in computer science from the Katholieke Universiteit Leuven (KU Leuven), Leuven, Belgium, in 2009. He is currently a Research Assistant at K.U. Leuven ESAT/ELECTA, where he is working toward the Ph.D. degree.

Sven Claessens received the M.Eng. degree in computer engineering from Stedelijke Industriële Hogeschool Antwerpen Mechelen (IHAM), Belgium, in 1991. He has been active as a software engineer in different domains like telecommunication, medical, micro-electronics and energy. Since 2009, he is employed at the Flemish Institute for Technological Research (VITO), and is involved in research projects related to Smart Grids.

Kevin Mets received the M.Sc. degree in computer science from Ghent University - IBBT, Ghent, Belgium, in 2009. He is currently a Research Assisent at the Internet Based Communication Networks and Services research group at the Department of Information Technology, Ghent University, where he is working toward the Ph.D. degree.

Chris Develder received the M.Sc. degree in computer science engineering and a Ph.D. in electrical engineering from Ghent University (Ghent, Belgium), in July 1999 and December 2003 respectively. From Jan. 2004 to Aug. 2005, he worked for OPNET Technologies, on transport network design and planning. In Sep. 2005, he re-joined the research group IBCN in the Dept. of Information Technology (INTEC) at Ghent University as a post-doctoral researcher, and as a post-doctoral fellow of the FWO since Oct. 2006. In Oct. 2007 he obtained a part-time, and since Feb. 2010 a fulltime associate professorship at Ghent University. His research interests include dimensioning, modeling and optimizing optical (grid/cloud) networks and their control and management, smart grids, as well as multimedia and home network software and technologies. He regularly serves as reviewer/TPC member for international journals and conferences (IEEE/OSA JLT, IEEE/OSA JOCN, IEEE/ACM Trans. Networking, Computer Networks, IEEE Network, IEEE JSAC; IEEE ICC, IEEE SmartGridComm, ECOC, etc.)

Geert Deconinck received the M.Sc. degree in electrical engineering and the Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven (KU Leuven), Leuven, Belgium, in 1991 and 1996, respectively. Since 1997, he has been with the staff of the Department of Electrical Engineering, K.U.Leuven, first as a Postdoctoral Fellow and now as full Professor. His research interests include the design, analysis, and assessment of software-based fault-tolerant solutions to meet real-time, dependability, and cost constraints for embedded applications on parallel and distributed systems.