MOST - Designing a vendor and technology independent toolkit for building monitoring, data preprocessing, and visualization

Robert Zach, Ardeshir Mahdavi

(Mag. DI (FH) Robert Zach, Department of Building Physics and Building Ecology, Vienna University of Technology, Karlsplatz 13, 1040 Vienna, Austria, robert.zach@tuwien.ac.at)
(Univ. Prof. Dr. Ardeshir Mahdavi, Department of Building Physics and Building Ecology, Vienna University of Technology, Karlsplatz 13, 1040 Vienna, Austria, amahdavi@tuwien.ac.at)

1 INTRODUCTION

Currently available building management systems (BMS) could be improved in view of a number potentially important functionalities. For example, real-time data access to salient – and often dynamically changing – building information (such as zone temperature and energy use) could be provided via appropriate data processing and performance modeling applications (such as spreadsheets, mathematical routines, simulation tools). Moreover, additional data processing functionalities could be offered, involving, for example, the calculation and display of performance data in a structured spatial and temporal manner. These improvements would facilitate the exploitation of the critical benefits that could result from the integrated and concurrent analysis of multiple building data streams (Raftery et al. 2010, O'Donnell 2009, Neumann and Jacob 2008). Such benefits include:

- Operation energy optimization through improved management of technical building systems.
- Increased awareness in building users regarding their impact on buildings' energy use.
- Early detection (and treatment) of deficiencies and malfunctions in energy systems and devices, thus effectively supporting a preventive maintenance regime.
- Successive building performance improvement and optimization via the analyses of dynamically updated building energy and performance data bases.
- Long-term accumulation of empirical information on buildings' energy and environmental performance toward improving the design, construction, and operation of existing and new buildings.

We argue that, due to the long life cycle of buildings and the fast evolution of building technologies, a vendor and technology independent approach for monitoring is essential to ensure future-proof comprehensive data collection. Furthermore, independence from the building market developments can be increased by using open-source technologies.

To facilitate data utilization for all interest groups, open software interfaces play an important role. Technical limitations (no real-time data access, missing interfaces for batch processing, etc.) can prevent processing applications from accessing building data in the required way. Unstructured measurement data sets can further increase analyses effort and reduce exploitation potential. Therefore, application independent interfaces and appropriate data preprocessing (calculation of timely structured data sets, linked queries of energy use for specific time intervals and building zones under specific occupancy conditions, etc.) is critical to maximize data usage for all interest groups (facility manager, occupant, building owner, etc.).

1.1 Approach

To achieve the desired building monitoring characteristics, the Monitoring System Toolkit - MOST - is presented. Based on five components (Connectors, Database-Core, Java-Framework, GWT based web interface, Matlab-Framework), the toolkit facilitates beneficial use of building data in various processing applications. It provides powerful preprocessing functions (e.g., generation of temporally structured data sets), offers interfaces for batch processing (MySQL, OPC-UA, etc.) and includes applications for data aggregation, display, visualization, and analysis (e.g. psychrometric and thermal comfort chart plots, data encapsulation and export, etc.).

To probe and gauge the utility of the toolkit in a realistic and practical context, two distinct buildings in Vienna, Austria were used for reference implementation. One is a new building, with some existing monitoring infrastructure elements. We illustrate how this building's data can be accessed using the toolkit, by connecting to all possible building automation infrastructures. The second building is forty years old and provides no reusable building automation infrastructure. Therefore, an independent system needed to be installed for monitoring.

2 MONITORING SYSTEM TOOLKIT

The software architecture of the proposed toolkit is shown in figure 1. It consists of *Connectors* to collect data from various building systems, a *Database* for historical data storage and data preprocessing, a *Java-Framework* which serves different software interfaces, a *Web Interface* for simple data access and visualization, and a *Matlab-Framework* to simplify complex data processing.

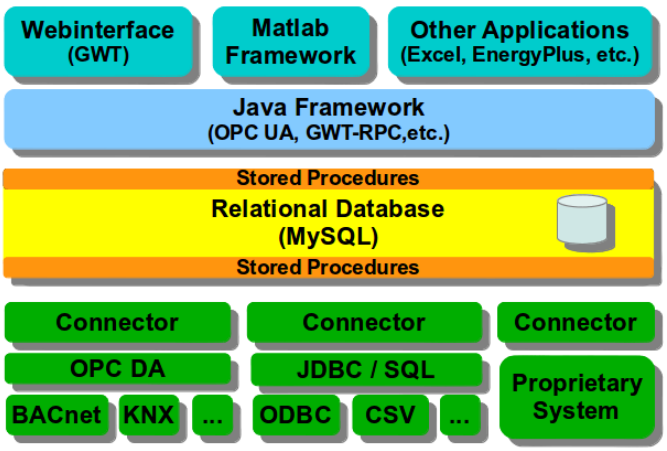

Fig. 1: Software architecture of MOST

2.1 Connector

To support vendor independent communication with common building management systems, two connectors (OPC DA and JDBC/SQL) were developed.

OPC DA Connector

The Open Process Control (OPC) Data Access (DA) connector allows connecting to common building automation networks such as BACnet, KNX, M-Bus, ZigBee, as well as to many building management systems (BMS). All technologies which provide OPC DA Server software can be accessed by the connector, as shown in figure 2. Since all OPC DA Servers provide the same Application programmable Interface (API), uniform communication to different building networks is possible.

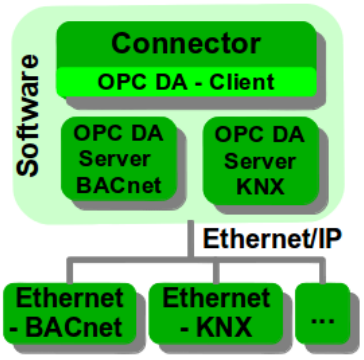

Fig. 2: OPC Client / Server infrastructure

OPC DA supports registration for events (value changed, etc.) at datapoints. This allows to process building data streams in real-time. Currently, the OPC Connector is implemented with the programming language Gamma, using the OPC Datahub (OPC Datahub 2012) environment. Figure 3 shows the class diagram of the actual implementation.

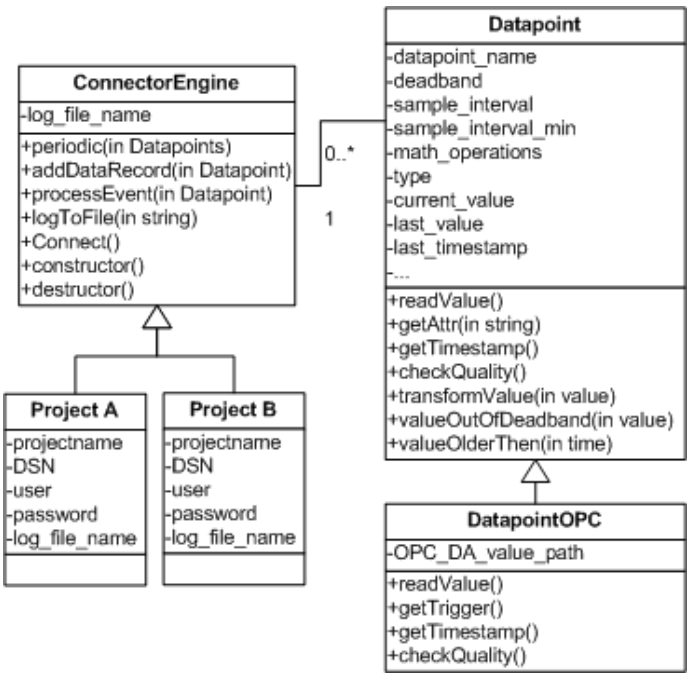

Fig. 3: Class diagram of the OPC DA Connector

Each project inherits from the *ConnectorEngine*. On startup, a *DatapointOPC* object is instantiated for each datapoint with an OPC data source defined in the MySQL database. Adaption of measurements within the Connector (e.g. to convert the measurement into a desired format) can be achieved by overwriting the *transformValue()* method. Because of the limited adaption options and vendor lock risks caused by using the proprietary environment OPC Datahub / Gamma scripting, a technology independent alternative is envisioned. Within the Google Summer of Code 2012 (GSOC 2012), a Java based OPC Connector (using the OPC DA library JEasyOPC 2012) will be developed.

JDBC/SQL Connector

The second connector allows communication with systems and read/write file formats, which are supported by Java Database Connectivity (JDBC) compatible drivers. It therefore enables data access to various databases (Oracle, Microsoft SQL, ODBC compliant databases, etc.) and popular file formats such as CSV, Excel, etc. based on the Structured Query Language (SQL). Due to technical limitations (the JDBC library is not notified when new data is added to the database/file), this connector supports data transfer by polling in a periodical manner only (i.e., every minute/hour/day/week/etc.). The JDBC Connector is implemented with the programming language Java. An extract of the class diagram is shown in figure 4.

The class *Datapoint* is a Data Abstract Object (DAO) of a datapoint in the MySQL database. The class *DpConnectorJdbc* implements the connector to any JDBC compatibe source. It automatically detects the table structure of the data source based on the information described in table 1. By probing the defined column names, appropriate SQL statements are generated. This enables support for most common data source structures shown in figure 5 (a) and (b). Support for additional data sources (e.g. proprietary systems) can be added by extending *the* class *DpConnector* and overwriting the method *getSourceData()*.

Since all communication of the presented connectors to existing building systems is based on Ethernet/IP, various installation setups are possible. For example, remote access to different building systems can be realized by using a Virtual Private Network (VPN). Figure 6 shows a configuration setup where the monitoring server collects data from various building systems (located anywhere in the world) through a VPN. By using an adaptive routing configuration on the VPN-GWs (e.g. OpenWrt 2012), infrastructure independent plug and play installation is possible. For example, the VPN-GWs can scan for possible Internet connections (Ethernet/IP, WLAN, UMTS, etc.) and automatically connect to the best fitting one. Based on the VPN a secure communication to the monitoring server is guaranteed. By using watchdog scripts on the VPN-GWs, automatic reconnection in case of communication faults can reduce data loss and maintenance effort.

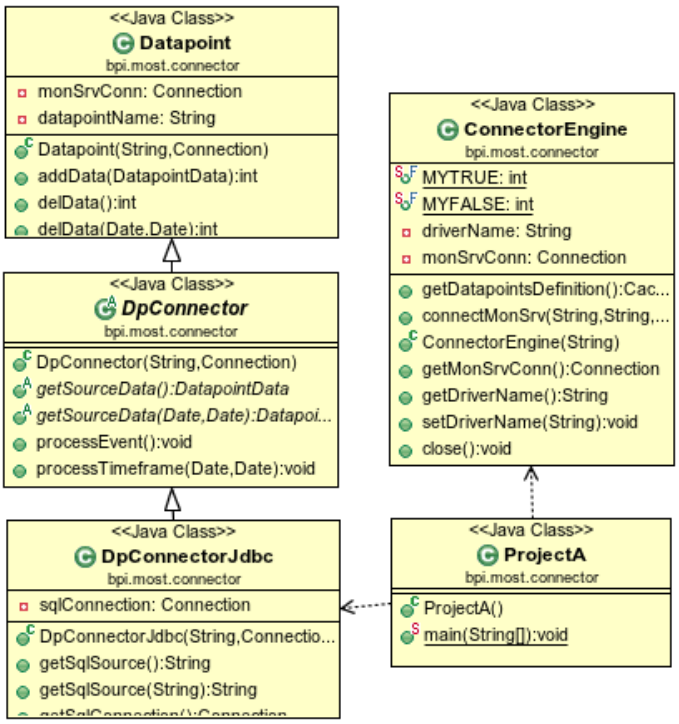

Fig. 4: Class diagram of the JDBC Connector

a) Multiple columns

Timestamp	Sensor 1	Sensor 2	etc.
2012-01-01-15:00	23,5	14,1	...
2012-01-01-15:10	22,3	14,4	...

b) Multiple rows

Timestamp	Sensor (ID)	Value
2012-01-01-15:00	Sensor 1	23,5
2012-01-01-15:00	Sensor 2	14,1
2012-01-01-15:10	Sensor 1	22,3
2012-01-01-15:10	Sensor 2	14,4
...	...	...

Fig. 5: Supported table structures of the JDBC Connector

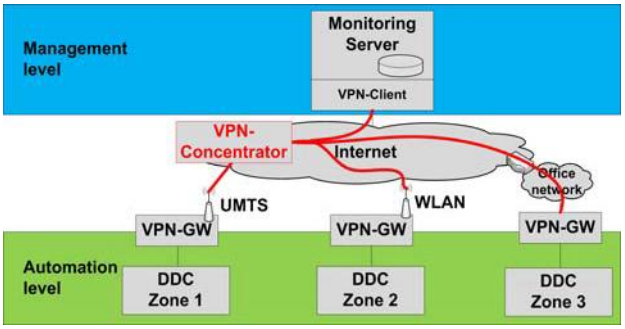

Fig. 6: Using a Virtual Private Network for monitoring over building limits

Datasource definition	Description
sqlTableName	Name of the table including the measurements
sqlTimestamp-Column	Columnname of timestamp
sqlDatapoint-Column	Columnname of the datapoint/sensor
sqlDatapointName	ID of the datapoint (required only if all datapoints are in the same column)
sqlValueColumn	Name of column containing the value (required only if all datapoints are in the same column)

Table 1: Variables for database structure detection

2.2 Database

To store historical data, the relational database MySQL is used. The database logic is completely decoupled from client applications by using stored procedures for all kind of data access. It supports the processing of measurements by just “dropping” them in the database. Data storage rules optimize performance and disk usage based on deadband, sample interval, and minimal sample interval parameters of each datapoint. Location management of physical sensors/actors and virtual datapoints is treated by logical grouping using zones. Based on the rich usage of stored procedures, performance and permission issues can be handled in a more fine-grained way than with direct access using SQL. It also enables a centralized implementation of data preprocessing algorithms. This can optimize data-query performance and prevents redundant code in different client applications (Matlab, Excel, EnergyPlus, etc.). Table 2 shows some stored procedures used to get periodic values with a desired interval for any kind of measurement-dataset. This allows different applications to access the measurements in a widely supported, simple, and uniform way. The processing application does not need to deal with the challenge of getting the measurements in the right format. Implementation details (Entity-Relationship Model, MySQL stored procedures, etc.) and performance benchmarks are explained in Zach et al. 2012.

Name	Description
getValuesPeriodic (*dpt, start, end, period*)	This stored procedure automatically selects and calls the required function (analog or digital) for the generation of values, depending on the type of the data point (*dpt*). The arguments *start* and *end* define the requested timeframe. *Period* specifies the time interval of the returned values. A quality index gives feedback about how many real measurements are available for the calculated values.
getValuesPeriodicAnalog (*dpt, start, end, period, mode*)	This stored procedure returns periodic values for any type of analog measurement (temperature, CO2, rhu, etc.). In the default *mode* a linear interpolation and arithmetic average is used for calculating periodic values. If the requested period contains more then one measurement, the arithmetic average is calculated. If no measurement is available for the requested period, a linear interpolation to the next measurement is done.
getValuesPeriodicBinary (*dpt, start, end, period, mode*)	This stored procedure returns periodic values for any type of digital measurement (window/door state, etc.). It supports three *modes*: - majority / sample & hold, - dominant "0" / default "1" and - dominant "1" / default "0". The majority/sample & hold mode returns the majority if more then one measurement is available in the requested period. If no measurement is available, the last value of the previous period is returned. The mode dominating "0"/default "1" returns "0", if one or more measurements in the requested period are "0". If no measurement is available, the default value "1" is returned. The mode dominating "1"/default "0" works the same way, but swaps "0" and "1".

Table 2: Data-preprocessing with stored procedures

2.3 Data-abstraction Java-Framework

MOST is separated in server side (Java-Framework) and client side (GWT) components as shown in figure 7. The server side is used as an abstraction layer for diverse building data streams and provides various software interfaces (OPC UA, GWT-RPC, etc.) for different processing applications (Webinterface, Excel, Matlab, etc.). To obtain information concerning parameters that are not directly measureable, so called "virtual datapoints" are introduced in the framework. A virtual datapoint can be, for example, the energy use of a zone, which represents an aggregation of all energy measurements in this zone. Generic virtual datapoints (energy use, number of people, etc.) are provided by the framework. Additional virtual datapoints can be added by extending the Java class *DatapointServer*. From the user's point of view, the same mode of access can be applied to both physical (sensor-based) and virtual datapoints. To provide building data with the widely supported software interface OPC Unified Architecture (UA), a generic information model was developed. The OPC UA information model uses the zone information defined in the MySQL database to generate the tree structure shown in figure 8. The implementation of the OPC UA interface is currently in an early development stage.

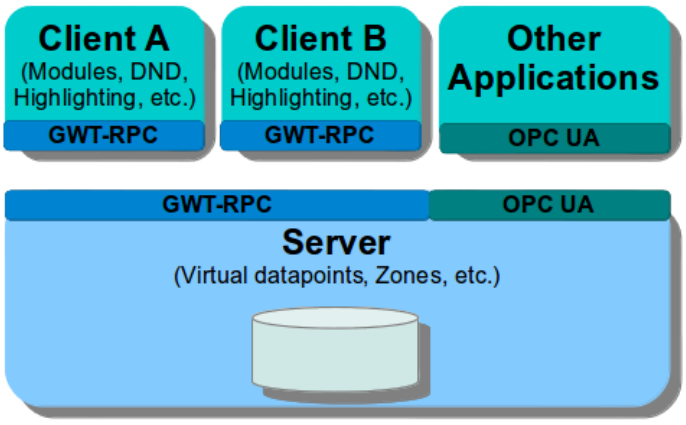

Fig. 7: Communication between server and client side

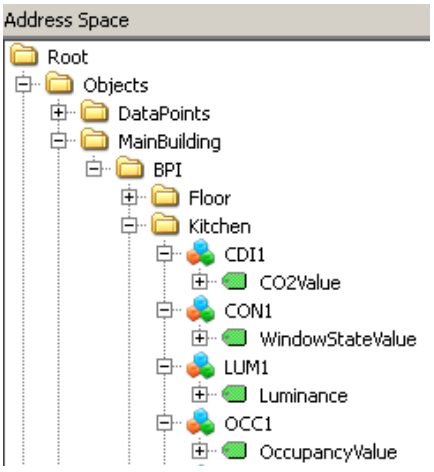

Fig. 8: Building data access using OPC UA

2.4 Web-Interface based on GWT

Based on the Java-Framework, a web interface is developed using the Google Web Toolkit (GWT). GWT enables web development with the programming language Java for server (running on a central server station) and client side code (running in the user's browser). Client side code is converted to platform optimized JavaScript at compile time. This strategy allows the development of reusable components for several use cases. Each use case is implemented using an independent module and has a main menu entry as shown in figure 9. A generic module implementation is available as a starting point.
Several reusable components are developed to simplify the implementation of new modules. The drag and drop functionality of GWT is enhanced with several features (highlighting of droppable areas, etc.) and various wrapper classes are provided for data visualization (*DragWindow*, *ChartWrapper*, etc.). To show the potential of the visualization framework, some modules covering various use cases were implemented. For example, the chart module allows creating trend charts from any datapoint by dropping the datapoint on the module. An export module is intended to be implemented to show how data can be requested based on various rules (e.g. only workdays, only defined time slots, etc.). Currently, data visualization possibilities within a two or three dimensional building model are being explored. Details of the Java-Framework (class diagram, virtual datapoints, etc.) and the web visualization using GWT (module based design, drag and drop support, etc.) are explained in Zach et al. 2012.

2.5 Data-processing Matlab-Framework

To simplify complex data processing with Matlab, a dedicated framework was developed. It provides common data processing algorithms in an object-oriented way. Figure 11 shows a cutout of the class diagram. The *MainCtrl* initializes the framework (instantiates required objects – Datapoint's, StatisticCtrl, etc.) and keeps track of the database connection. A set of Objects (*StatisticCtrl*, *PlotCtrl*, *DatapointCtrl*, *Datapoint*, *ZoneCtrl*, etc.) provide various methods to simplify data analysis. Figure 10 shows a sample psychrometric chart generated with the method *plotMollierOfZone()*. Due to the open-source nature of the framework, functionality can be easily enhanced or adopted to respective needs.

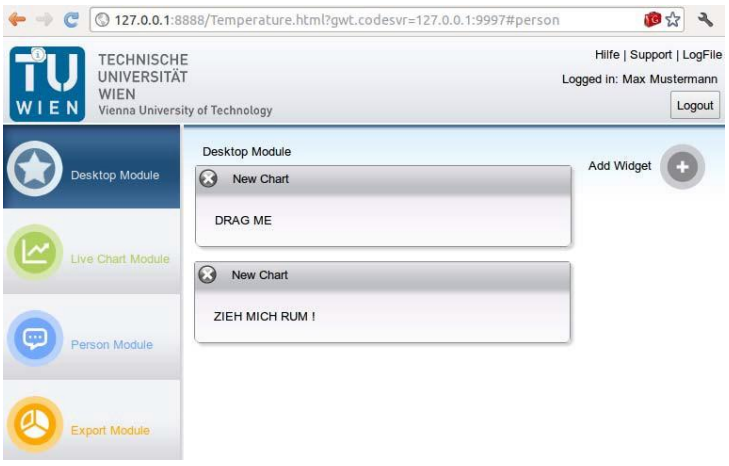

Fig. 9: Module based user interface design

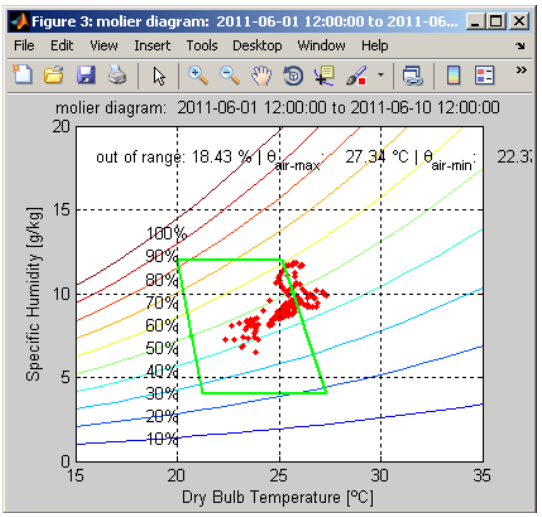

Fig. 10: Exemplary psychrometric chart generated with the Matlab-Framework

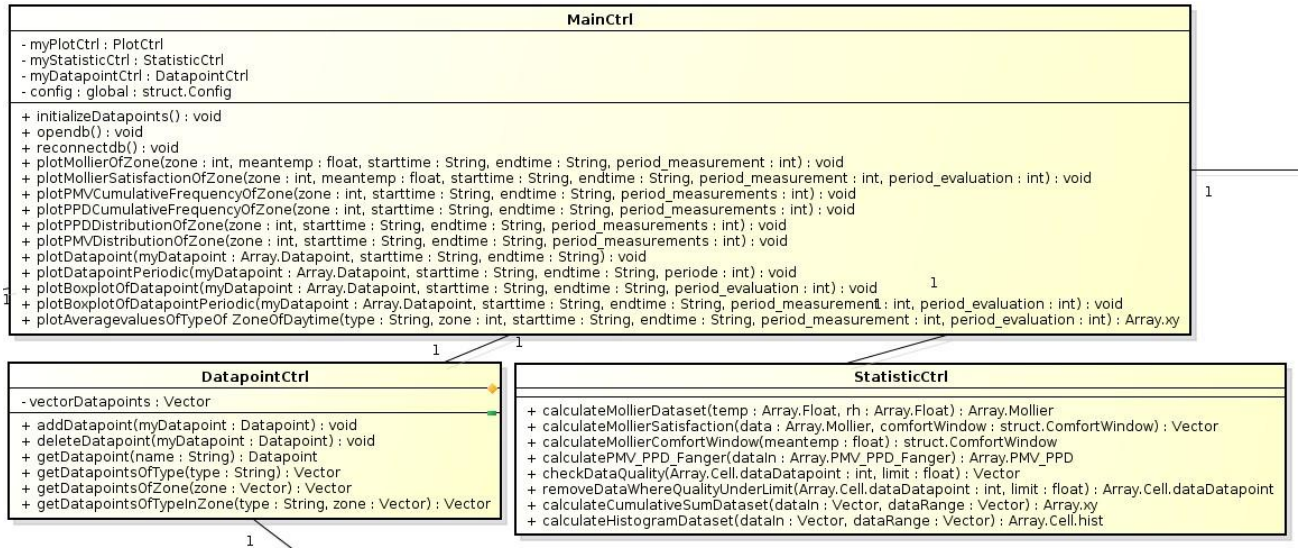

Fig. 11: Cutout of the Matlab-Framework class diagram

3 PROTOTYPICAL IMPLEMENTATION

To study real-life implementation and application scenarios, two buildings were selected and partly equipped with necessary monitoring infrastructure. These buildings house a number of offices, labs, and lecture rooms of the Vienna University of Technology. One of the buildings (Lehartrakt) was completed 2010 and provides reusable building automation infrastructure to various degrees. The second building (Mitteltrakt) was built more than 100 years ago and provides no reusable building automation infrastructure. As such, these buildings are representative of a large number of existing buildings in Vienna. They entail a wide range of technical challenges that need to be met in order to realize the postulated dynamic data acquisition and processing architecture in the context of existing buildings. Such challenges pertain specifically to the technology update requirements for incorporation of high-resolution sensory and metering capabilities, device connectivity, and cross-platform data transfer.

3.1 Lehartrakt

Recently completed (2010), Lehartrakt is equipped to various degrees with current building automation technologies. Therefore, the monitoring system can reuse some of the sensor and network infrastructure to reduce installation efforts. Figure 12 shows a four layer model of the entire monitoring infrastructure. Many sensors in the existing BACnet, KNX, and M-Bus networks are reused. Electricity meters are added to the M-Bus system. All other sensors are added with the wireless fieldbus EnOcean to reduce installation costs.

All building automation networks (KNX, M-Bus, EnOcean) are accessed with the OPC DA connector. Heating ventilation and air conditioning is controlled by a building management system which provides a MySQL interface only. Therefore the JDBC connector is used to poll required data. Relative humidity, room air temperature, window/door states, and occupancy are monitored with self-powered sensors using the wireless fieldbus EnOcean. Carbon dioxide and volatile organic components are measured with legacy sensors, which are equipped with EnOcean wireless modules. Electrical energy use is measured with M-Bus meters. Light and blind states are monitored by tapping the KNX fieldbus. All measurements are tunneled to the monitoring server using a VPN concentrator.

3.2 Mitteltrakt

The building Mitteltrakt provides no reusable building automation infrastructure at all. To reduce installation costs, a fully independent wireless approach for the fieldbus network is adapted. EnOcean was chosen as the wireless fieldbus system because of its optimized design for low power use. It allows the construction of self-powered sensor devices, which reduces installation efforts and increases installation flexibility. Nevertheless, some sensors still need a power supply due to the energy use of the sensor technology. For example, a CO2 sensor needs a significant power supply to drive its heating coil. Many sensors with integrated EnOcean communication technology are available on the market. Missing sensors can be easily developed using legacy devices and EnOcean-modules of the STM and TCM series.

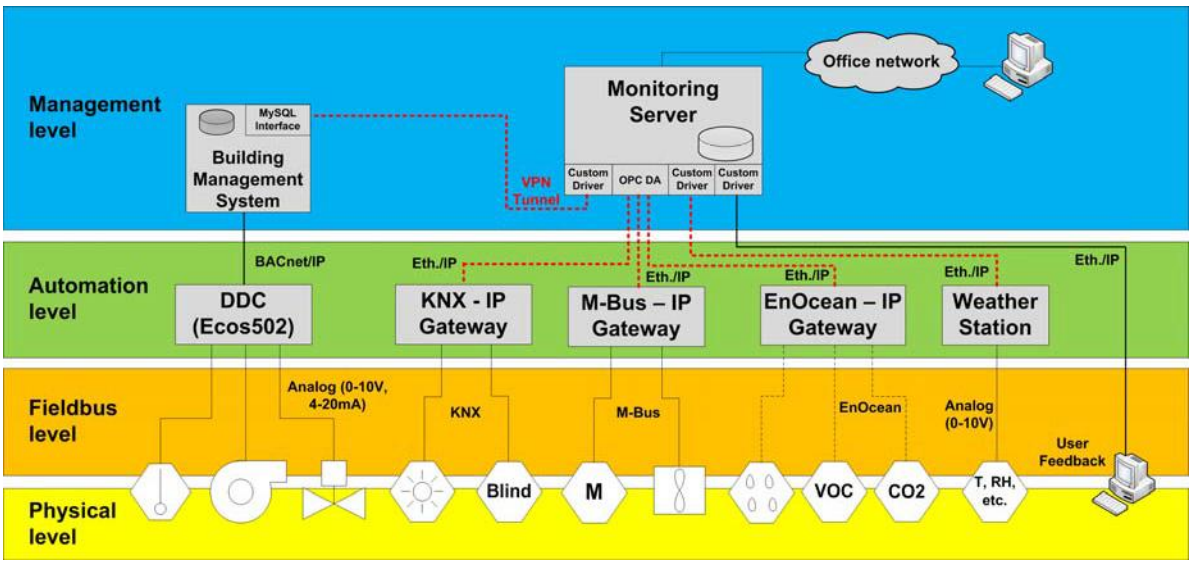

Fig. 12: Four layer model of the monitoring setup in Lehartrakt

4 DISCUSSION AND CONCLUSION

By using the proposed toolkit, multiple salient data streams originating from a building's operation can be analyzed in a comprehensive manner. Thus, the envisioned use cases can be realized. Provided interfaces enable real-time data processing independent of software. Powerful data preprocessing methods are implemented that support effective data analysis. Such methods involve, for example, linked queries of energy use for specific time intervals and building zones under specific occupancy conditions. The scalable design of the monitoring framework can accommodate very different system requirements and supports new buildings (with existing building automation components) as well as independently conceived and implemented monitoring systems that can be realized in buildings without reusable infrastructure. The framework developed and the associated applications support building operators to rapidly respond to occupants' requirements. The toolkit is shown to be flexible, as demonstrated via successful implementations in two very distinct reference buildings.

Future research and development challenges involve the implementation of a software independent OPC DA connector to extend supported building systems, improvement of the OPC UA interface to increase compatibility with processing applications, enhancement of the web visualization to simplify usage for non-technical skilled users, and demonstration of various use cases (e.g. automatic calibration of simulation models, simulation-based fault detection, remote data collection for a number of projects, etc.).

5 ACKNOWLEDGEMENT

The research presented in this paper is supported by funds from the program “Innovative Projekte” of the Vienna University of Technology as well as from the Austrian Science Foundation (FWF): Project: I 545-N22. Additional support was provided by the division "Gebäude und Technik" (Amtsdir. Hodcek), which supplied us with real-world test beds. Information on further developments regarding the proposed monitoring toolkit is available on http://most.bpi.tuwien.ac.at.

6 REFERENCES

GSOC 2012. Google Summer of Code, February 2012, *http://code.google.com/soc/*

JEasyOPC 2012. Java based OPC DA Client, February 2012, *http://jeasyopc.sourceforge.net/*

Neumann C. and Jacob D. 2008. *Guidelines for the evaluation of building performance*. Freiburg, Germany: Fraunhofer Institute

O'Donnell J. 2009. *Specification of Optimum Holistic Building Environmental and Energy Performance Information to Support Informed Decision Making*. Doctorate, University College Cork, Ireland.

OPC Datahub 2012. OPC Toolkit, February 2012, *http://www.opcdatahub.com/*

OpenWRT 2012. A Linux distribution for embedded devices, February 2012, *https://openwrt.org/*

Raftery P., Keane M., O’Donnell J., Costa A. 2010. *Energy Monitoring Systems: value, issues and recommendations based on five case studies*. 9 – 12 May, Antalya, Clima 2010, International Conference on Sustainable Energy Use in Buildings.

Zach R., Schuss M., Bräuer R. and Mahdavi A. 2012. *Improving building monitoring using a data preprocessing storage engine based on MySQL*. 25 – 27 July, Island

Zach R., Glawischnig S., Appel R., Weber J., Mahdavi A. 2012. *Building data visualization using the open-source MOST framework and the Google Web Toolkit.* 25 – 27 July, Reykjavik, Island