www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

# A DISTRIBUTED AND SCALABLE APPROACH TO BUILDING MONITORING

SUBMITTED: November 2014
REVISED: December 2014
PUBLISHED: January 2015 at http://www.itcon.org/2015/12
GUEST EDITORS: Mahdavi A. & Martens B.

***Robert Zach***
***Department of Building Physics and Building Ecology, Vienna University of Technology, Austria***
***robert.zach@gmail.com***

***Harald Hofstätter***
***Department of Building Physics and Building Ecology, Vienna University of Technology, Austria***
***harald.e259.hofstaetter@tuwien.ac.at***

***Christian Tauber***
***Department of Building Physics and Building Ecology, Vienna University of Technology, Austria***
***christian.tauber@tuwien.ac.at, http://bpi.tuwien.ac.at/***

***Ardeshir Mahdavi***
***Department of Building Physics and Building Ecology, Vienna University of Technology, Austria***
***bpi@tuwien.ac.at***

***SUMMARY:*** *The present contribution describes a scalable approach to vendor and technology independent building monitoring and data processing. Powerful data pre-processing algorithms, virtual data points, automated building model calibration and various software interfaces are implemented across different software modules. The distributed software architecture enables scalable processing of desired data streams for various applications from the building to the urban level.*

***KEYWORDS:*** *building monitoring, building management system, building automation, building simulation*.

***REFERENCE:*** *Robert Zach, Harald Hofstätter, Christian Taube, Ardeshir Mahdavi (2015). A distributed and scalable approach to building monitoring. Journal of Information Technology in Construction (ITcon), Special Issue: ECPPM 2014, Vol. 20, pg. 159-172, http://www.itcon.org/2015/12*

## 1. INTRODUCTION

Existing monitoring solutions and building management systems (BMS) could be enhanced with regard to a number of potentially important functionalities. Data pre-processing functionalities could be offered, involving, for example, the calculation of data in a structured temporal manner (e.g. periodic values). Virtual datapoints could be used to provide non-measured data streams (e.g. thermal comfort within a zone, overall energy use of a zone, etc.) based on related sensor measurements. Different software interfaces could enable various data processing applications (Excel, MATLAB, BMS, etc.) to batch process required data streams. These advancements 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:

- Minimization of energy use through improved management of technical systems.
- Better informed building users regarding their impact on buildings' energy use.
- Facilitating a preventive maintenance regime via early detection of deficiencies and malfunctions in energy systems and devices.
- Continuous building performance improvement and optimization via the analyses of dynamically updated building energy and performance databases.
- Long-term accumulation of empirical information on buildings' energy and environmental performance toward improving the design, construction, and operation of existing and new buildings.

The present contribution proposes a modular monitoring infrastructure to collect and process building measurements (energy use, comfort parameter, etc.), which can be optimized in order to meet the requirements of each individual application. The proposed algorithms and design patterns are implemented in an Open-Source toolkit, called the Monitoring System Toolkit (MOST 2014, http://most.bpi.tuwien.ac.at). The toolkit facilitates beneficial use of building data in various processing applications. It provides powerful pre-processing functions (e.g., generation of temporally structured data sets, virtual data points, etc.), supports different data storage technologies (e.g. MySQL, Cassandra, Neo4j), offers interfaces for batch processing (oBIX, OPC-UA, custom RESTful web service, etc.) and includes applications for simulation model calibration, data aggregation, display, visualization, and analysis (see Figure 1).

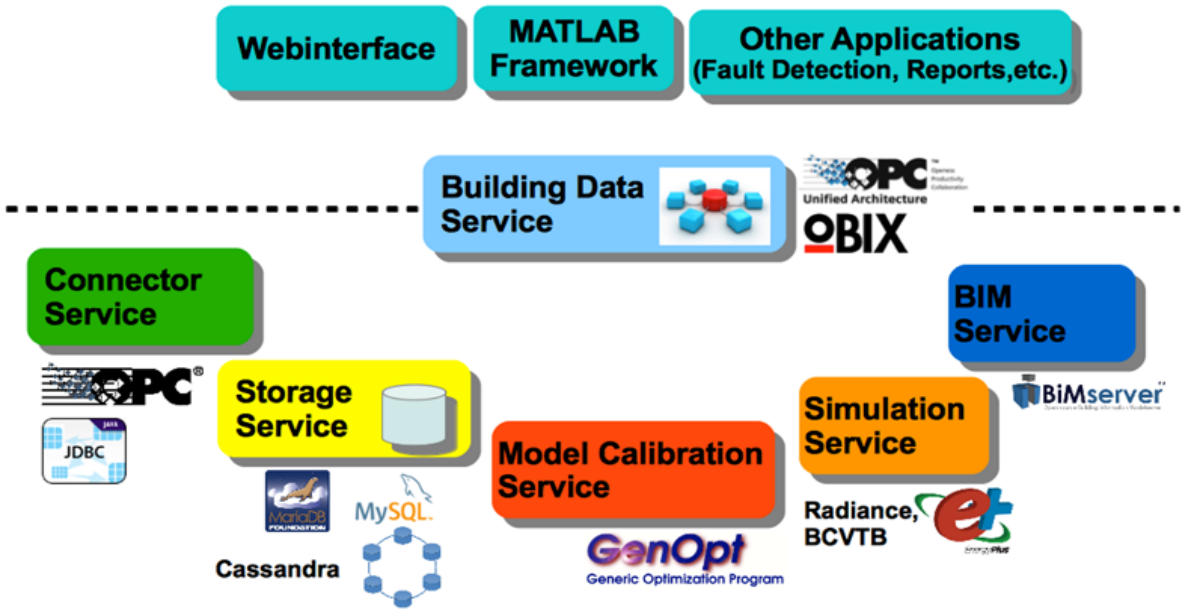


*Figure 1. Services provided by the Monitoring System Toolkit (MOST)*

## 2. APPROACH

Monitoring systems in general and building monitoring systems like MOST in particular have to (i) collect, (ii) store and (iii) pre-process data before (iv) providing access to it. These seemingly trivial steps increase in complexity as soon as different data sources, pre-processing strategies, storages and data access interfaces are required. To build such a system in a stable, reliable, maintainable and extensible way it is vital to separate the system into distinct parts addressing multiple concerns. Therefore MOST is composed of several modules, communicating only over well-defined interfaces. Each module is responsible for a particular task like collecting, storing, pre-processing or providing data.

Modules communicate with each other only via well-defined interfaces, hiding implementation details. This fact allows to transparently interchange one implementation with another as long as the interface remains. Other modules do not notice the change because only implementation was swapped, but importantly, not the interface. This behaviour can be used to enable different module-implementations and deployment-strategies, which are optimized for distinct use cases.

## 3. MONITORING SYSTEM TOOLKIT

MOST collects measurements from various data sources (sensors/actors, BMS, weather forecast, etc.), stores historical data, enables various preprocessing algorithms and provides desired information with different software interfaces (oBIX, OPC UA, etc.). Each point of interest is called a "*data point*" and can provide data from a physical sensor/actor or calculated by different algorithms (virutal data point – formula based, simulation based, etc.). This results in a generic building data interface, which provides values (measurements) as tuples of type <timestamp, value> belonging to a data point. Furthermore, MOST stores information regarding the association of data-sources and data points, grouping of data points, etc. Future work could get this metadata from Building Information Models (BIM). BIM information could be imported from a file (e.g. import of an IFC file) or integrated as a service (requesting information from a BIM-server on demand). To enable a clean separation of building metadata and runtime building measurements, BIM standards as the Industrial Foundation Classes (IFC) or Green Building XML are required to include attributes to link sensors in the BIM model (e.g. IfcSensor) to the respective data point in the monitoring system. The architecture of MOST is composed of different modules as illustrated in Figure 2. Data is either stored into or read from MOST.

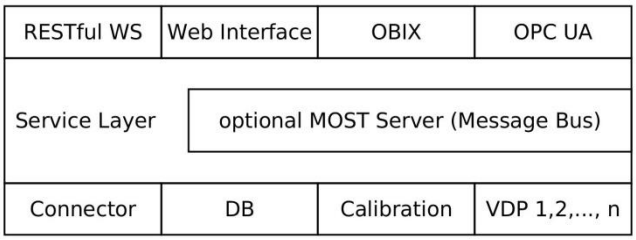


*Figure 2. Software architecture of MOST modules*

The connector module reads measurements from sensors and stores it in the database. All requests go through the service layer, which delegates it to the correct module. Historic measurements are provided from the database module. Requests to virtual data points are calculated by the respective virtual data point module. Different software interfaces are provided with the modules OPC UA, oBIX, and RESTful WS to support many processing applications (Excel, MATLAB, BMS solutions, etc.). The web interface module includes a prototypical implementation of a web visualization. The module calibration enables automated calibration of a building simulation model (such as EnergyPlus 2014), which, for example, can be used for a virtual data point (to compute non measured data).

## 4. POSSIBLE QUERIES

Different data points will produce data at different points in time. Sensors send their data periodically (every 5 minutes) or only when values change (e.g. event triggered contact sensors). To allow reasonable analysis and comparison of data point values, data pre-processing algorithms are required. For example window state information (contact sensor) is stored, marking the window as open or closed. Subsequent data processing

applications may want to obtain this information in a periodic manner (e.g., hourly). Therefore, the pre-processing algorithm must deliver, for each discrete interval, either the value "open", or "closed". This is achieved via appropriate reasoning depending on the use case of the processing application. For example, a window may be declared open or closed if a corresponding action took place in the respective interval. Alternatively, a window may be declared open if it was open during most of the respective interval. To account for this and other data pre-processing challenges, a number of data pre-processing algorithms are developed. Figure 3 to Figure 7 show examples of data requested with the method *get-ValuesPeriodic(dp, start, end, period, mode)* using different modes. Crosses mark stored measurements while circles show calculated return values. *Dp* is the id of the requested datapoint. *Start* and *end* define the requested timeframe. *Period* contains the desired interval in seconds. *Mode* enables data pre-processing with different algorithms.

In mode 1, a linear interpolation and temporally weighted arithmetic averages are used for calculating periodic values. If the requested period contains more than one measurement, the temporally weighted arithmetic average is calculated. If no measurement is available for the requested period, a linear interpolation to the next measurement is performed. Mode 2 uses sample & hold instead of linear interpolation. Mode 3 (majority/sample & hold) returns the majority of non-zero (true) or zero (false) values if more than one measurement is available in the requested period. If no measurement is available, the last value of the previous period is returned. Mode 4 (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. Mode 5 (dominating "1"/default "0") works the same way, but swaps "0" and "1". By using the proposed data pre-processing algorithm and virtual data points, processing applications can query desired information in the required data structure. This enables the processing application to focus on the respective use case.

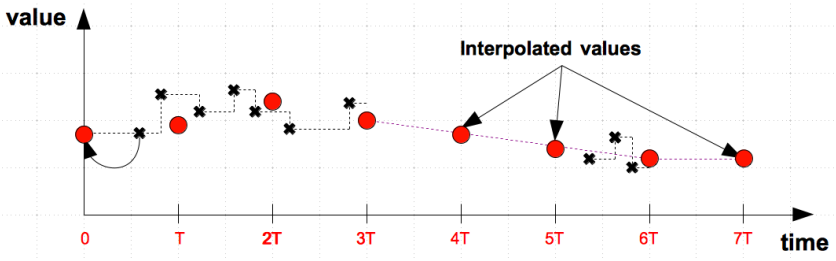


*Figure 3. Data pre-processing - mode 1: time-weighted average / linear interpolation*

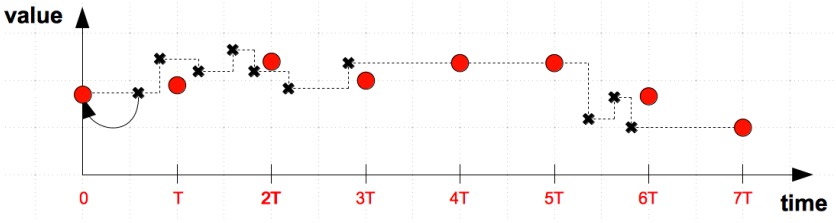


*Figure 4. Data pre-processing - mode 2: time-weighted average / sample & hold*

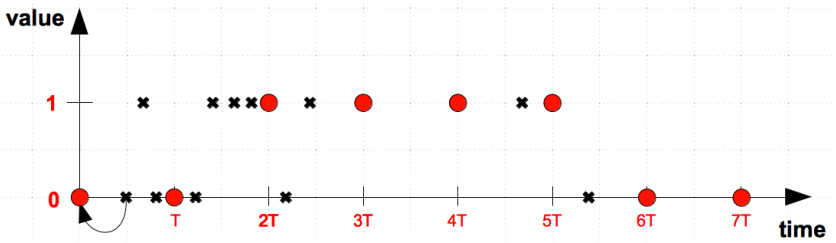


*Figure 5. Data pre-processing - mode 3: majority decision / sample & hold*

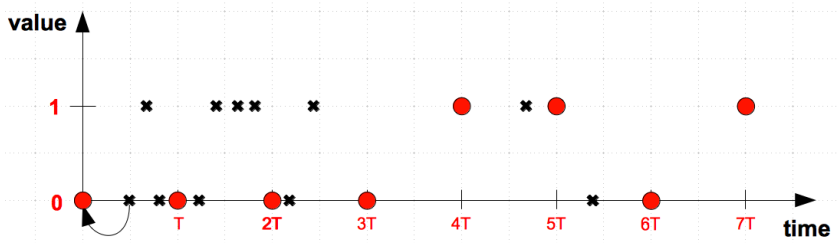


*Figure 6. Data pre-processing - mode 4: forced 0 / default 1*

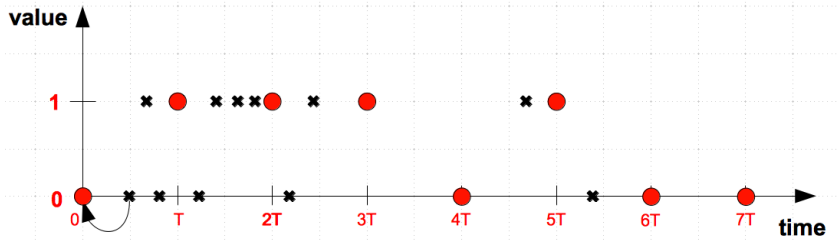


*Figure 7. Data pre-processing - mode 5: forced 1 / default 0*

## 5. MOST MODULES

Table 1 provides an overview of MOST's currently available modules.

*Table 1 Overview of currently available modules*

| *MOST-Module Name* | *Description* |
|---|---|
| Connector (most-connector) | Driver for different data-source to MOST |
| MySQL (most-mysql) | Handles database access for meta-data like Data points and Zones. It optionally supports the storage of pure measurement for small sized deployments. |
| Neo4j (most-neo4j) | Handles database requests for data point measurements stored in Neo4j. |
| Cassandra (most-casandra) | Handles database requests for data point measurements stored in Cassandra. |
| Calibration (most-calibration) | Calibrates a building simulation model (currently EnergyPlus only) in an automated and periodic manner. |
| Virtual Datapoint (most-vdp) | Contains several implementations of so-called virtual data points, which provide access to not directly measured data. |
| MOST Server (most-server) | Routes requests between different MOST modules. Only required for distributed deployments |
| REST (most-rest) | Exposes MOST data with a RESTful web service. |
| OPC UA (most-opcua) | Exposes MOST data through OPC Unified Architecture. |
| Obix (most-obix) | Exposes MOST data through oBIX. |
| Web (most-web) | Provides an out of the box web application to query, visualize and export monitored data. |

## 5.1 Connector Module

The target of the module connector is to collect measurements from various sensors, fieldbus technologies and building management systems (BMS). Currently support for the following Technologies is implemented. (i) JDBC compatible data sources and therefore most relational databases (MySQL, Microsoft SQL, Oracle, etc.) and structured file formats (e.g. CVS). This way access to many BMS is supported. (ii) OPC DA data sources, which enables access to most sensor and fieldbus technologies (KNX, BACnet, LonWorks, M-Bus, etc.). (iii) EnOcean USB 300 supporting low cost monitoring with a Raspberry Pi and energy harvesting sensors (Zach et al. 2014).

Zach et al. 2012 describes the proposed data collection approach in more detail. Future work could be based on the project IoTSyS (Jung et al. 2012), which allows native access to several building automation technologies like BACnet, KNX, ZigBee, DALI and Lon Works.

## 5.2 MySQL Module

To enable historical data access compatible modules with different persistence strategies are evaluated and implemented in the modules (i) most-mysql, (ii) most-cassandra and (iii) most-neo4j.

At the moment the module MySQL is mandatory as it stores BIM related information (e.g. Zones, Attributes of Sensors, etc.). Additionally it realizes one of currently three backends to store pure data point measurements. Previous work (Zach et al. 2012) has shown the limits of storing measurements in a relational database like MySQL. Therefore, two alternative storage solutions based on NoSQL are evaluated, namely Cassandra (Apache Cassandra 2014) and Neo4J (Neo4j 2014).

## 5.3 Neo4j Module

Neo4J is a graph database (Indrawan-Santiago 2012), where data is stored as graph. Vertices are called nodes and edges are called relationships in Neo4J. Both are able to store additional data in so called properties. Hierarchical data like BIM information and its associated data can be stored very naturally as graph in Neo4J. Foreign keys of the MySQL schema are transformed to explicit relationships between entities. Instead of joining to tables to link associated data, you have to traverse from one node through relationship to its neighbors. Neo4j's distributed implementation and scaling capabilities made it a reasonable candidate for storing data point measurements. In the future it may be used for storing hierarchical data like zones and data points too because this simplifies queries on recursive associations.

## 5.4 Cassandra Module

The second NoSQL storage we are evaluating is the column family (Indrawan-Santiago 2012) database Apache Cassandra. Designing a Cassandra-based model starts with the query domain, instead of starting with the domain model (Wang & Tang 2012). Therefore we analyzed required data queries for MOST and built a scalable Keyspace utilizing shards to distribute data across multiple nodes. Measurements of each data point and each month are stored in an independent set of data described by the Keyspace shown in Figure 8. Multiples of these datasets can be stored on individual nodes in the Cassandra cluster. The distribution of this datasets is done randomly. By splitting data in time and source in a random way, a linear scalability of the overall storage cluster is reached. The Cassandra module of MOST distributes and collects requested data from respective nodes, transparent for the client applications. Further details can be found in Glawischnig et al. 2014.

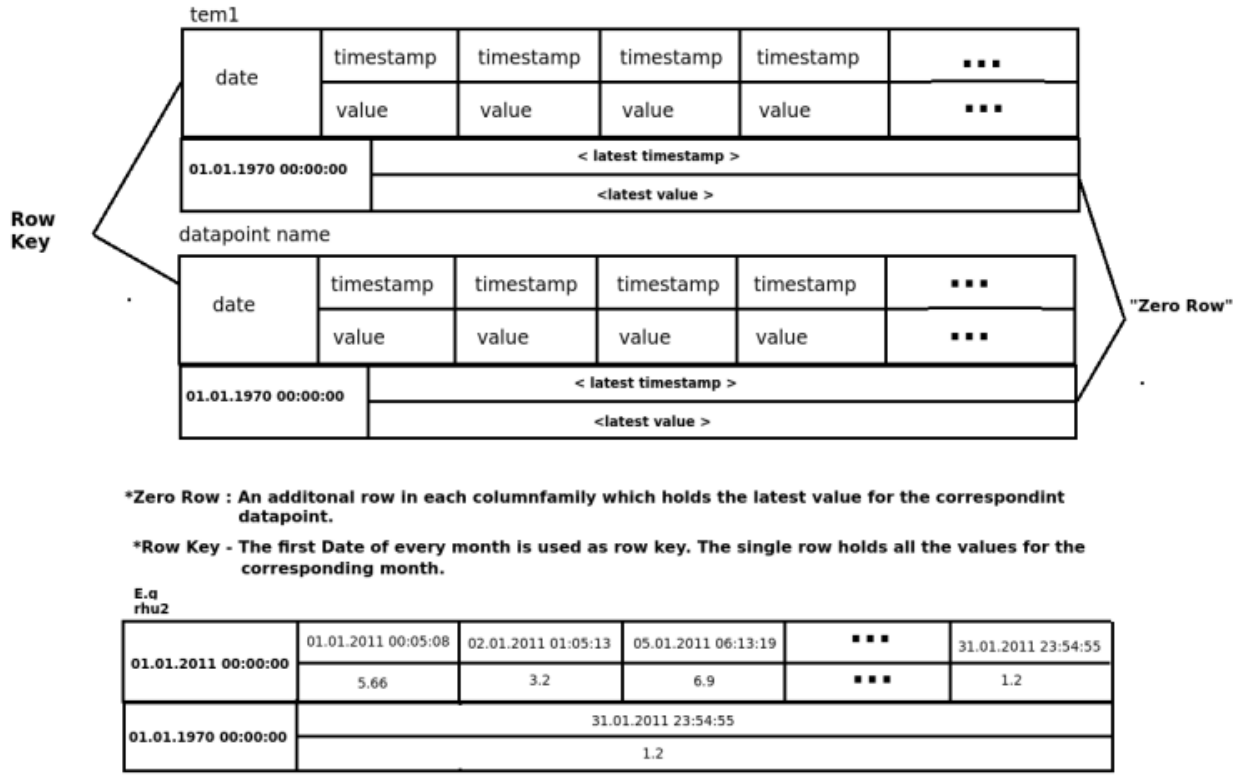


*Figure 8. Keyspace of the MOST Cassandra Modul*

## 5.5 Calibration Module

This module enables the automatic calibration of a building simulation model based on measurements stored in MOST. By periodically calibrating the simulation model with the latest measurements various applications are possible (virtual data points based on simulation tools, simulation based fault detection, etc.).

The proposed layer-based structure of the calibration module allows simple component exchangeability and expandability, including implementations for different simulation and optimization tools. The current implementation extends GenOpt 2014 (version 3.1.0). The native version of GenOpt calculates the cost function for the optimization process within the respective simulation tool. Technology issues make it very complicated to implement cost functions within the simulation software if they depend on values from external data sources (e.g. measurements from a building monitoring system). For this reason, the cost function calculation was moved from the individual simulation software to the optimization domain (GenOpt). This was done by extending GenOpt with a plugin-interface, which enables the implementation of the cost function in the optimization domain. The plugin-interface provides the output paths from the simulation tool used and requires from respective implementations to return the calculated cost function value.

Figure 9 shows the layer-based structure of the proposed implementation with EnergyPlus 8.2 as currently used simulation tool. The MOST Calibration Service is schematically separated in:

1) Pre-processing: contains jobs such as the creation of a weather file for the calibration period;
2) Calibration: includes the driver for the optimization program;
3) Post-processing: contains any tasks that should be done after a calibration, e.g. deployment of the calibrated model.

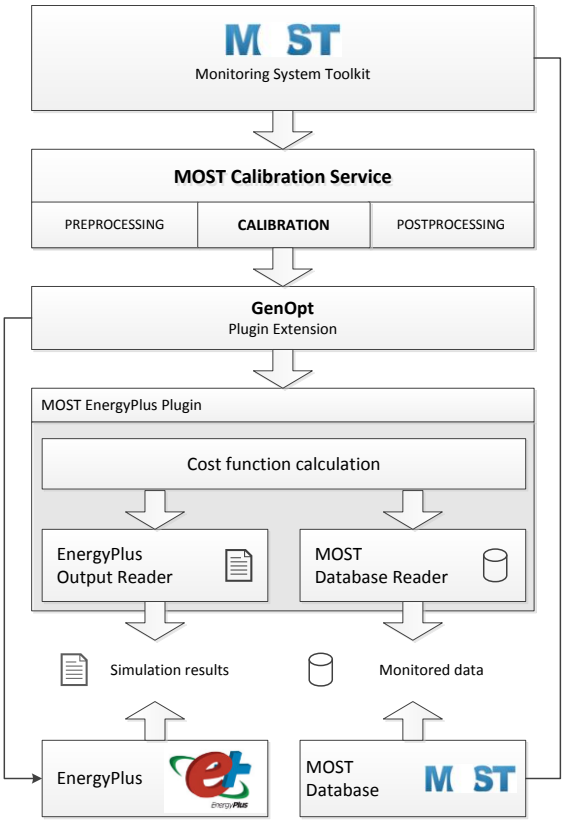


*Figure 9. Proposed layer-model using the example of the Monitoring System Toolkit and EnergyPlus.*

In the current implementation optimizes an EnergyPlus simulation model based on measurements from the Monitoring System Toolkit. It is further separated in cost function calculation, currently realized as RMSD calculation, and the basic data reader objects. This layer approach shows how components could be exchanged easily, for example when switching to another simulation tool without having to adjust the whole solution. In this case a new custom reader object has to be implemented that could provide the simulation output to the cost function algorithm in a standardized way.

The target of the overall calibration process is to optimize certain variables in the building simulation model. These variables can be classified into two fundamental types:

- Time independent variables, which display no (or negligible) change in the regarded time period (e.g. U-Value of a window).
- Time dependent variables, which are defined with an array of values over time, known as schedules (e.g., state of a window: open/closed).

Time-dependent variables involve more complexities in model calibration because an optimum of a series of values has to be found to reach optimum model performance. A less computationally expensive approach could be populating the model with a pool of possible schedules for time-dependent variables, where the best-fitting one is selected in the calibration process. To provide this pool of schedules in a simple manner, a number of daily schedules can be picked randomly from the relevant monitored data of a certain period of time. Each set of schedules pertaining to occupant presence and interactions with building systems should be obtained from the same day to avoid inconsistencies in the schedules. Alternatively, occupants' presence and behavioural models, trained with the monitored data, can be used in order to generate occupancy-related profiles, among other things, based on environmental factors. Tauber et al. 2014 describes these issues in more detail.

## 5.6 Virtual Data point Module

This module can provide data, which is not directly measured with a physical sensor. Currently the following examples of virtual data points (VDP) are evaluated.

- A virtual data point, which takes the mean surface temperature of a radiator and calculates its heating power based on geometric information of the radiator and the surrounding room temperature.
- A virtual data point wrapping the MOST domain specific language (most-DSL) implemented in Scala. It enables users to weave datapoint's values into mathematical expressions where particular values are evaluated at runtime based on the requested timeframe for evaluation. An expression computing the average temperature in °C of two data points "*tem1*" and "*tem2*" would be written as follows:

  ```
  (dp("tem1") + dp("tem2")) / 2
  ```
- Integrating most-DSL as VDP allows nesting an arbitrary graph of most-DSL expressions, whereas loops are not allowed. Assuming the last expression would be accessible as the VDP "avgTem", we could build a new VDP converting the result to °F:

  ```
  dp("avgTem") * 1.8 + 32
  ```
- A prototype of a simulation based virtual data point is currently in development. Using the simulation tool EnergyPlus it calculates data, which is not directly measured (e.g. thermal comfort). A calibrated simulation model is used to provide reliable simulated results.

## 5.7 MOST Server Module

As discussed later MOST can be deployed in a distributed fashion, so that different modules are running on different machines. The MOST Server's core capability is to provide facilities for inter-module communication in form of a Message oriented Middleware (MoM) resulting in loosely coupled components. Modules register its communication channel at the server at startup, so that they can communicate with each other via message passing. Consequently all communication flows through the Most Server's provided MoM. Data point's values can be observed by subscribing to a data point's "OBSRV" topic as described in Glawischnig et al. 2014. Virtual data points share a queue per type, that means all virtual data points from type "mostDsl" listen to the same queue so that incoming messages are taken by at most one instance. For simpler implementation we created helper classes to hide the asynchronous nature of message passing. This makes it easy to develop new components without paying too much attention to communication details.

## 5.8 REST Module

This module provides data access via a simplified REST service over HTTP with XML/JSON-marshalling. For example, to retrieve data from data point "tem1" the following URL can be called:

```
http://your.server.com/most-
rest/v1/dp/tem1/data?from={UCTdatetime}&to={UCTdatetime}
```

For periodic data the required period in seconds and an optional mode are part of the query parameters:

```
http://your.server.com/most-
rest/v1/dp/{name}/periodicdata?from={UCTdatetime}&to={UCTdatetime}&period={
integer}&mode={integer}
```

Every system able to perform HTTP requests and XML/JSON manipulation is able to integrate requested via this interface. This ranges from websites, to Office Software (e.g. Microsoft Excel), to Business Process Management Suits.

## 5.9 OPC Unified Architecture Module

Based on the subproject opcua4j (Hofstätter 2012), the module OPC Unified Architecture (UA) server was developed. Zone information is used for the OPC UA address space. A zone connects to its data points with a

*hasComponent* references. Attributes such as a data point's unit are represented as OPC UA properties. This module enables data access for all processing applications supporting the OPC UA interface.

## 5.10 oBIX Module

The module oBIX provides data access based on the Open Building Information eXchange (oBIX 2014) standard. This standard provides data with the default contracts *Points*, *Alarm* and *History*. To support the proposed pre-processed queries a new contract was developed.

## 5.11 Web Module

In addition to the REST, OPC UA and oBIX interface that allows machine-to-machine communication, there exists a ready-to-go rich web application (RIA) written in Java (GWT), deployable in any Servlet container. The application acts as simple user interface for endusers and serves two use cases for: data visualization and data export. Visualization is realized by charts where values from one or more data points are shown for a user specified time intervals. Data points may be hard to find if one does not know their names. Therefore we introduced 3D visualization based on a BIM model (IFC) to illustrate a datapoints location and simplify search. Interaction is mainly realized via drag and drop gestures to support natural behavior. While dragging an object, only droppable areas are highlighted to guide the user and increase usability (Zach et al. 2013). Regarding data export, it is possible to export one or more data point values in a given time interval into CSV files. This allows users without much technical know-how, to query required data and process it further in any application supporting CSV import. A screenshot of the web interface is shown in Figure 10.

*Figure 10. Prototypical web interface for accessing data from the Monitoring System Toolkit-MOST*

# 6. INTER-MODULE COMMUNICATION

All introduces modules communicate with each other only via well-defined Java interfaces. These interface are mapped on a Message oriented Middleware (MoM) as shown in Figure 11. This way, diverse modules can be transparently moved to different computer hardware. Some modules (e.g. the Virtual Data point Modul) can even be deployed multiple times. On user request, the instance with the shortest queue will be used for processing. Figure 12 shows a typical user request. The most server recognizes that the requested data ("avgTem") comes from a virtual data point of type "mostDSL". It finds an instance providing a virtual data point of type "mostDSL" and forwards the request. The requested data is calculated and sent back to the client.

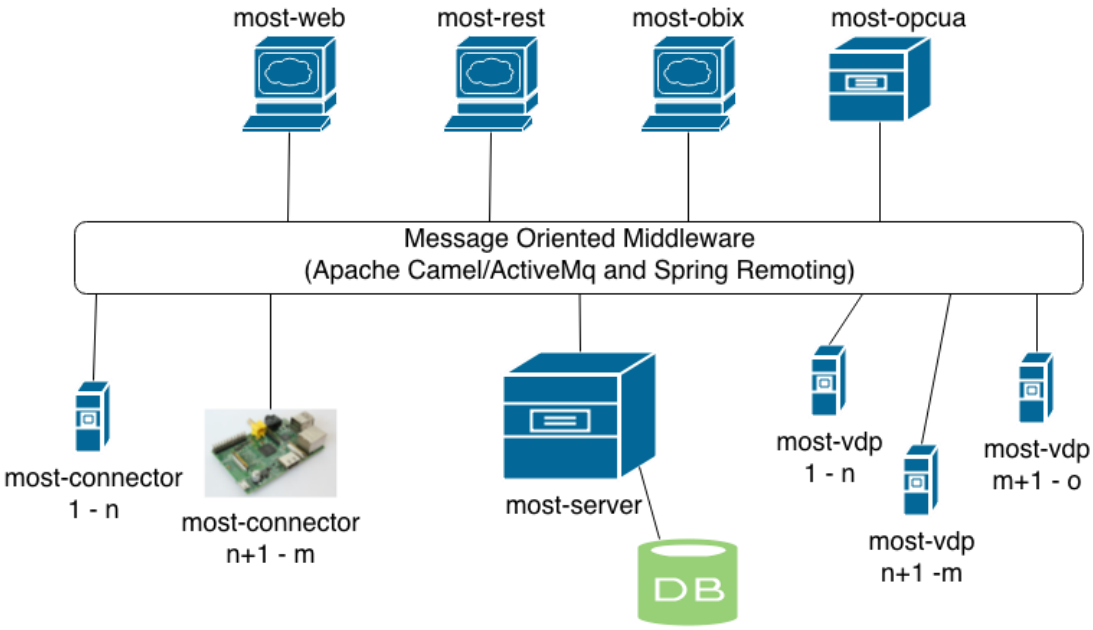


*Figure 11. Inter-module communication based on a Message oriented Middleware*

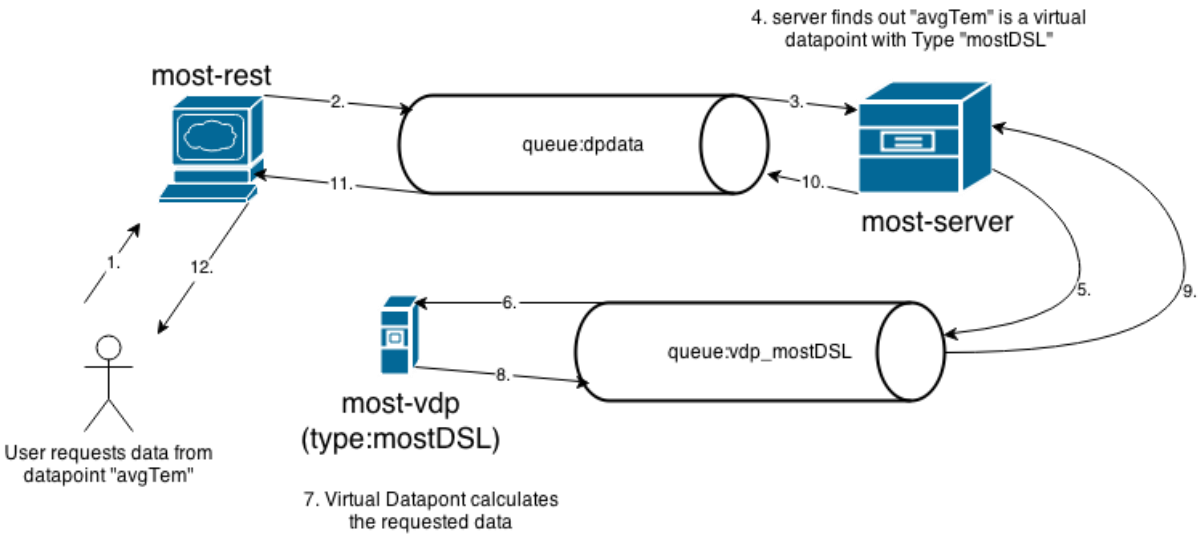


*Figure 12. Typical request to a virtual data point of type "mostDSL"*

## 7. POSSIBLE DEPLOYMENT SETUPS

With the proposed modular approach MOST's deployment options range from a single embedded system (Zach et al. 2014) to a multi-server high performance setup. This is possible due to the fact that MOST supports a middleware (message bus) between the modules. For example, MOST can be deployed on a low cost (i) single machine (Figure 13) or within a scalable (ii) distributed setup (Figure 14). The first option is reasonable for simple deployments, where the whole MOST application is packaged into a single runnable and deployed on a single machine. In this configuration, the service layer realizes inter-module communication via direct method calls between modules. In a distributed setup, all modules are connected to the MOST Server's MoM to communicate with each other. The MOST Server creates communication channels at runtime, based on the number of connected modules. This allows distributed and redundant deployment of components (see VDP 2 is deployed two times on different machines in Figure 14) as well as adding/removing parts during runtime.

Figure 15 and Figure 16 show how the proposed framework can be applied in a real world monitoring setup.

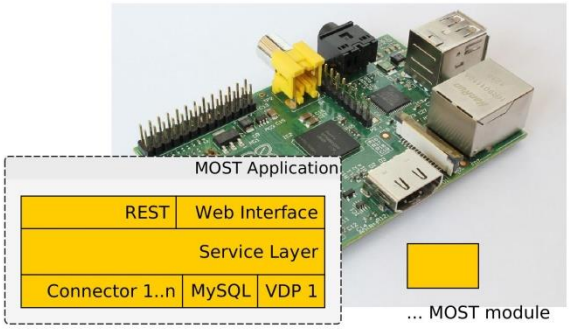


*Figure 13. MOST deployed on an Raspberry Pi*

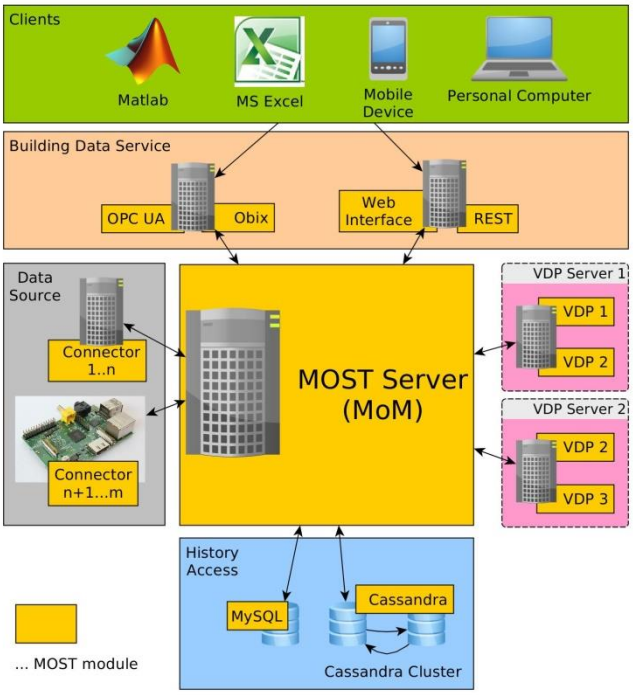


*Figure 14. MOST modules deployed on several distributed machines*

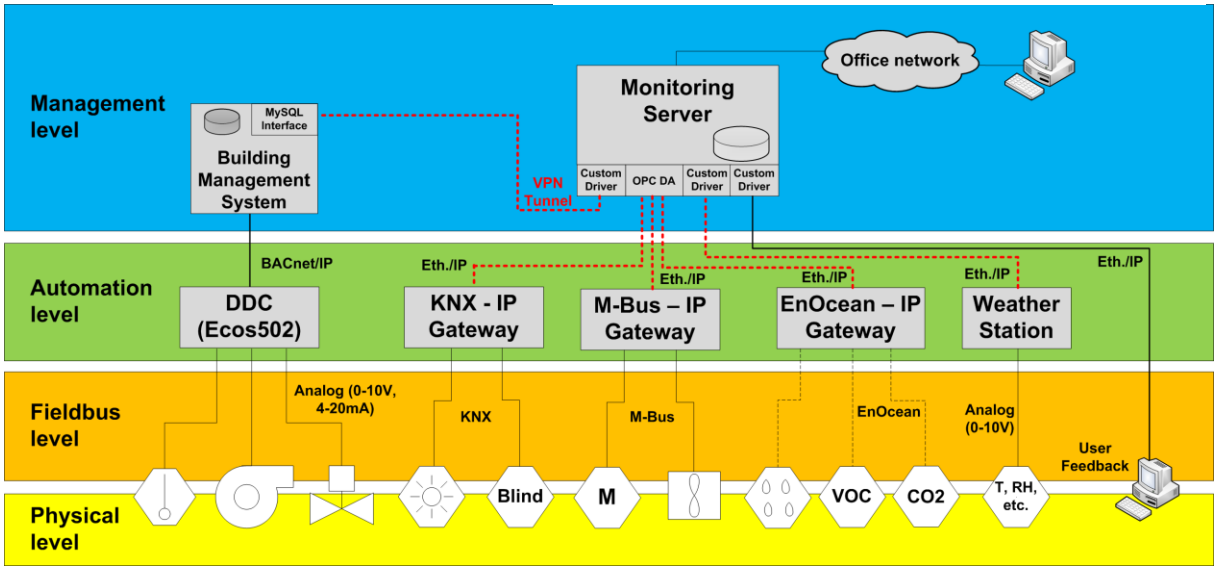


*Figure 15. Four layer model of a comprehensive monitoring setup in the building "Lehartrakt" of the University of Technology Vienna*

*Figure 16. Some sensors installed in the building "Lehartrakt"*

## 8. CONCLUSION AND FUTURE OUTLOOK

The proposed pre-processing algorithms and the support of virtual data points enables processing applications to query desired information in the required data structure (e.g. hourly data of the overall energy use in zone X). Client applications can access building data in a uniform way independent of the installed sensor, fieldbus and BMS technology. By supporting access with different software interfaces in real-time, batch processing in various client application is possible. These steps significantly simplify the challenge of processing building related data and enables reuse of processing applications (Figure 17).

Creating a domain specific language for MOST is a novel approach to describe data evaluation expressions without much technical knowledge. Wrapping the most-dsl with a virtual data point facilitates deep integration in the system.

Future work could include further integration of simulation based virtual data points, extending the MOST connector with additional technology support and the integration of BIM technologies. Improvements in stability can be accomplished by keeping track of active components and recognize partial failures of the system. Dynamic (un)registration of modules needs to be monitored by the MOST Server. Performance analysis is crucial to test scalability of the proposed system.

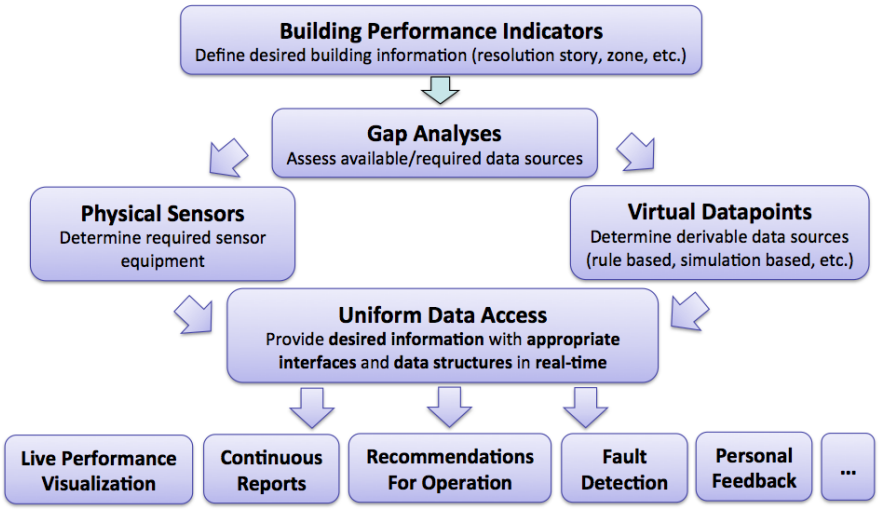


*Figure 17. Proposed approach for building monitoring*

## 9. ACKNOWLEDGMENT

The research presented in this paper is supported by funds from the "Klima- und Energiefonds" within the program "Neue Energien 2020". Since MOST is an Open-Source project, a number of people have been involved. Namely, Prof. Ardeshir Mahdavi, Robert Zach, Harald Hofstätter, Rainer Bräuer, Stefan Glawischnig, Christian Tauber, Reinhard Zach, Paul Alexander, Michael Hönisch, Regina Appel, Jakob Korherr, Alexej Strelzow, Christoph Lauscher, David Bittermann, Christoph Kaltenriner and Michael Leichtfried.

## 10. REFERENCES


Apache Cassandra. (2014). NoSQL database, *http://cassandra.apache.org/*

EnergyPlus. (2014). Building Performance Simulation Tool, *http://apps1.eere.energy.gov/buildings/energyplus/.*

Glawischnig S., Hofstätter H., and Mahdavi A. (2014). A distributed generic data structure for urban level building data monitoring. ICT-EurAsia 2014, Bali, Indonesia

GenOpt. (2014). GenOpt – generic optimization program. Available at: *http://simulationresearch.lbl.gov/GO/*

Hofstätter H. (2012). opcua4j - open source implementation of an opc ua server in java. Available: *https://code.google.com/p/opcua4j/*

Indrawan-Santiago M. (2012). Database research: are we at a crossroad? Reflection on NoSQL. NBiS 2012

Jung M., Weidinger J., Reinisch C., Kastner W., Crettaz C., Olivieri A., and Bocchi Y. (2012). A transparent IPv6 multi-protocol gateway to integrate building automation systems in the internet of things. IThings 2012.

MOST (2014). Monitoring system toolkit. *http://most.bpi.tuwien.ac.at*

Neo4j (2014). Graph based database. *http://www.neo4j.org*

Neumann C. and Jacob D. (2008). Guidelines for the evaluation of building performance. Freiburg, Germany: Fraunhofer Institute for Solar Energy Systems.

Tauber C., Tahmasebi F., Zach R. and Mahdavi A. (2014). *Automated simulation model calibration based on runtime building monitoring.* ECPPM Vienna

oBIX. (2014). Open Building Information eXchange. *http://www.obix.org*

O'Donnell J. (2009). Specification of optimum holistic building environmental and energy performance information to support informed decision making. Dissertation, University College Cork, Ireland.

Raftery P., Keane M., O'Donnell J. and 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.

Wang G., Tang J. (2012). The NoSQL principles and basic application of cassandra model. CSSS 2012

Weber J., Zach R., Tahmasebi F. and Mahdavi A. (2012). Inclusion of user-related monitoring data in the run-time calibration of building performance simulation models. in: BauSIM 2012 - Gebäudesimulation auf den Größenskalen Bauteil, Raum, Gebäude, Stadtquartier, C. Nytsch-Geusen et al. (Ed.); IBPSA Germany-Austria, 1 (2012), Paper-Nr. 164, 7 p.

Zach R., Glawischnig S., Hönisch M., Appel R. and Mahdavi A. (2012). *MOST:* An open-source, vendor and technology independent toolkit for building monitoring, data preprocessing, and visualization. in: "eWork and eBusiness in Architecture, Engineering and Construction", G. Gudnason, R. Scherer et al. (Ed.); Taylor & Francis, (2012), ISBN: 978-0-415-62128-1; P. 97 - 103.

Zach R., Schuss M., Bräuer R. and Mahdavi A. (2012). Improving building monitoring using a data preprocessing storage engine based on MySQL. in: "eWork and eBusiness in Architecture, Engineering and Construction", G. Gudnason, R. Scherer et al. (Ed.); Taylor & Francis, (2012), ISBN: 978-0-415-62128-1; p. 151 - 157.

Zach R., Glawischnig S. and Mahdavi A., (2013). Advanced building data visualization using the monitoring system toolkit. in: CLIMA 2013 - 11th REHVA World Congress and the 8th International Conference on Indoor Air Quality, Ventilation and Energy Conservation in Buildings, K. Kabele, M. Urban, K. Suchý, M. Lain (Ed.); Society of Environmental Engineering (STP), 1/1/1 (2013), Paper-Nr. 283, 9 p.

Zach R., Paul A., Zach R. and Mahdavi A. (2014). Plug and play building monitoring: the potential of low cost components. in: Proceedings of the 10th European Conference on Product and Process Modelling (ECPPM2014), Vienna, Austria, 17-19 September 2014, A. Mahdavi, B. Martens, R.J. Scherer (Ed.); Taylor & Francis - Balkema, 1/1/Boca Raton|London|New York|Leiden (2014), ISBN: 978-1-138-02710-7; p. 265 - 270.