INCORPORATION OF RUN-TIME SIMULATION-POWERED VIRTUAL SENSORS IN BUILDING MONITORING SYSTEMS

Robert Zach, Stefan Glawischnig, and Ardeshir Mahdavi

Department of Building Physics and Building Ecology,
Vienna University of Technology, Austria

ABSTRACT

The present contribution describes an approach to implement simulation-powered virtual sensors in a building information framework. Measurements from physical sensors are used to calibrate simulation models. Subsequently, virtual sensors can derive information on parameters that are either difficult or expensive to measure. The presented building data service provides a uniform interface to virtual and physical sensors.

INTRODUCTION

Previous research and development efforts by the authors and other groups have shown the potential benefits that could result from the integrated and concurrent analysis of multiple data streams in buildings (energy use, indoor climate, etc.). For instance, within the framework of a recent research effort (MOST 2012, Zach 2012), we conceived and implemented a vendor and technology independent toolkit for building monitoring, data processing, and visualization. This monitoring toolkit provides a set of versatile preprocessing algorithms (e.g., to generate temporally structured data sets), offers interfaces for batch processing (MySQL 2012, OPC UA 2012, RESTful web service, etc.) and includes applications for data aggregation, display, visualization, and analysis (e.g., temperature distribution maps, psychrometric charts, thermal comfort plots, data encapsulation and export, etc.). The overall toolkit is implemented as open-source and available at http://most.bpi.tuwien.ac.at.

Experiences in this project have compelled us to further explore and implement the concept of run-time simulation-powered virtual sensors. Due to various reasons pertaining to technical constraints and cost considerations, there are limits to the deployment of real sensors in view their numbers, types, quality, locations, accessibility, connectivity, etc. Simulation, if reliably and effectively incorporated, can not only expand the reach of real sensors by creating additional virtual sensors, but could also support the pervasive and continuous monitoring of compound performance criteria (such as visual glare indices or various measures of thermal comfort) that could be otherwise monitored – if at all – under unreasonable costs.

APPROACH AND OVERVIEW

In the present paper, we first describe a general solution for the incorporation of such simulation-powered virtual sensors in the general architecture of a vendor and technology independent monitoring system. Thereby we outline the required underlying abstractions and techniques. Moreover, we demonstrate how, by integrating virtual sensors into the proposed building monitoring solution, processing applications (visualization, fault detection, predictive building systems control, etc.) can access diverse building related data in a uniform way. For the purposes of this demonstration, we show two prototypical implementations of virtual sensors.

BUILDING DATA SERVICE

The aforementioned building monitoring system consists of a number of software services, which enable the user to request desired building information. The general system structure is shown in *Figure 1*. The connector service collects data from diverse building systems or other data sources (e.g. personal feedback). The storage service provides historical data access to pure measurements. Information about the sensor location, measurement unit, accuracy, etc. is covered within the Building Information Model (BIM) service. The model calibration service periodically calibrates desired building simulation models (e.g. EnergyPlus 2012, Radiance 2012, etc.) based on monitored data (see Tahmasebi and Mahdavi 2012). The simulation service abstracts different simulation tools to a uniform interface. The building data service provides a uniform interface to all underlying data sources. It implements diverse software communication standards (e.g. OBIX 2012, OPC UA, RESTful Web Service, GWT-RPC 2012, etc.) to support a wide range of possible data processing applications. Provided software interfaces enable application to batch process building data in real-time. Prototypical applications are shown in Zach et al. 2012a (e.g., Webinterface, MATLAB Framework, etc.). The presented services are implemented as maven modules. Internal communication is based on Java-RMI 2012.

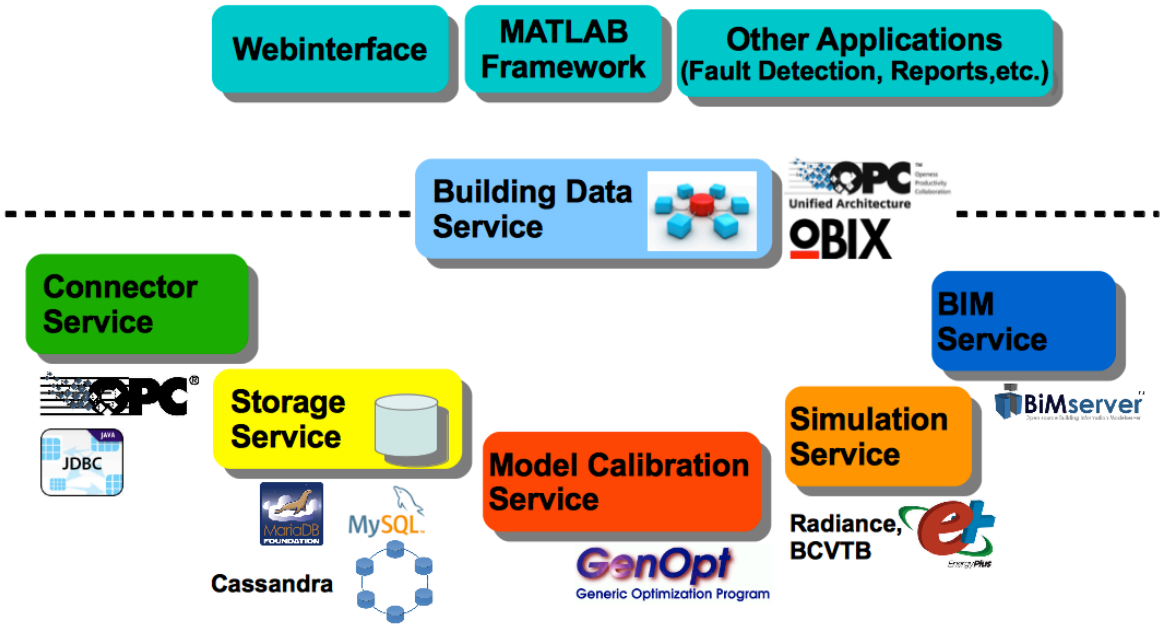

Figure 1. General structure of the proposed building monitoring system

The building data service contains so called virtual datapoints, which enable aggregation of real measurements to the desired information. They are used to obtain information that is not directly measured. The proposed implementation represents all information points with the abstract class *Datapoint*. *Figure* ***2*** shows a class diagram of how this datapoints can be accessed within the framework. Two distinct implementations of the class *Datapoint* are available (*DpPhysical* and *DpVirtualXXX*). The class *DpPhysical* represents real sensors and actors of a building and reads/writes directly to the storage service. Virtual datapoints are named with the prefix "*DpVirtual"*. To add a new virtual datapoint, the abstract class *DpVirtualFactory* must be implemented and registered to the Java Service Loader (by using the text file META-INF/services/bpi.most.server. DpVirtualFactory). Each implementation of a virtual datapoint has to provide a unique identifier (String ID). It needs to return an instance of the desired *Datapoint* implementation with the method *getVirtualDp(String uniqueString)*. On request, the *DpController* searches for the respective implementation and returns the desired instance.

Generic virtual datapoints are provided within the framework (see Zach 2012). Virtual datapoints can implement any kind of data processing functionality. Tests of prototypical implementations have been conducted with rule/formula based and simulation based virtual datapoints. For example, the formula based virtual datapoint *DpVirtualRadiatorHeatPower* calculates the heat power of a radiator based on the mean temperature and the standard heat output of the radiator together with the room temperature of the respective zone. The calculation is based on the K values described in DIN 1994. The simulation based virtual datapoint *DpVirtualRadiance* is currently under development. It calculates the visual conditions at any location (e.g. desktop) via Radiance. Each luminaire is modelled in the simulation model as an individual light source. The building data service provides state information (light on/off) for each luminaire. Given a request for data, all luminaire states within the simulation model are replaced with the associated values provided by the monitoring system. The sky model is generated based on the weather station measurements (irradiance data). Finally, Radiance is executed and the output is parsed and returned. In the current setup, all electrical lighting circuits are captured via the monitoring system. In practice, the state of luminaires may be generated in terms of another virtual datapoint, which could use occupancy (or other correlating factors) to make a pertinent assumption. Future developments will also include a sky scanner to generate a more detailed sky model.

Caching

To improve the performance of virtual datapoints, caching strategies can be applied. For example, the calculated values of a desired request can be stored using the storage service. The virtual datapoint checks on the next request if previous calculated values exist. New calculations are only executed for new requests. This strategy can significantly improve the performance while enabling virtual datapoints to include any calculation/simulation to retrieve the desired information.

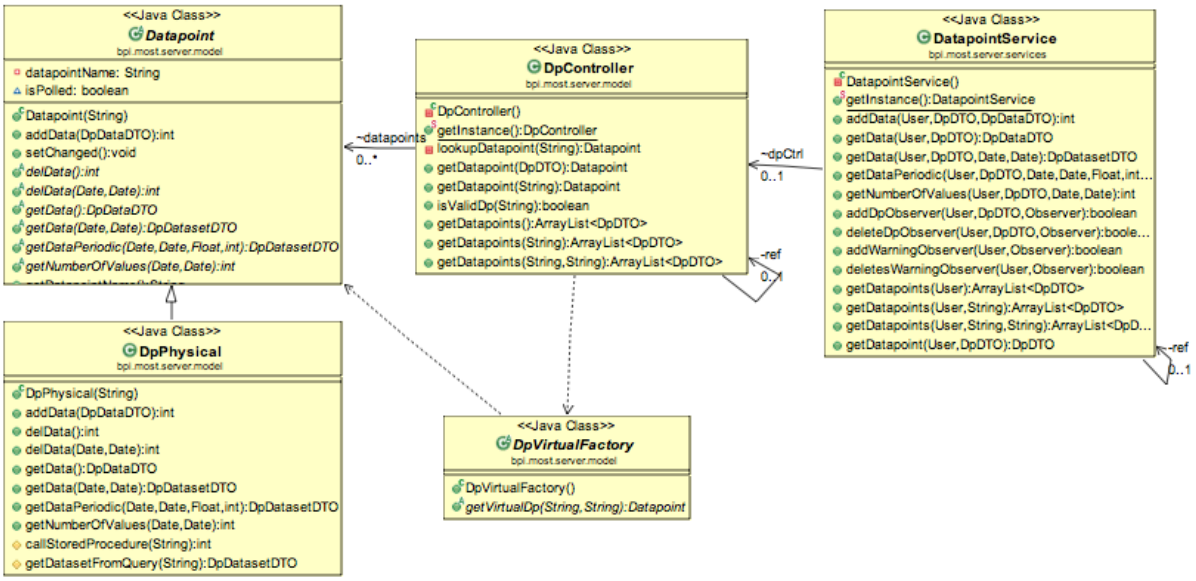

Figure 2. Class diagram of the virtual datapoint implementation

Depending on the amount of data supported by the storage service, different caching strategies may be appropriate. From the user's point of view, the same mode of access (abstract class *Datapoint*) can be applied to both physical (sensor-based) and virtual datapoints. The functionality provided by virtual datapoints enables the reduction of the vast amount of measured data in buildings to the information required. By using the data preprocessing functionality of the proposed storage service, information processing is accelerated. For example, the calculation of temporally structured data sets, e.g. hourly/weekly/monthly/etc. values, linked queries of energy use for specific time intervals and building zones under specific occupancy conditions, etc. are implemented within the storage service. This approach increases the performance and scalability of the overall system.

DATA PREPROCESSING

To simplify data analysis, powerful data preprocessing algorithm for data requests were developed. For example, window state information can be stored in a high-resolution fashion by saving timestamps marking the window opening and closing actions. Subsequent data processing applications may want to obtain this information in a periodic manner (e.g., hourly). Therefore, the preprocessing algorithm must deliver, for each discrete interval, either the value "open" or "close". This is achieved via appropriate reasoning depending on the use case of the processing application. For example, window may be declared open or close if a corresponding action took place in the respective interval. Alternatively, window may be declared open if it was open during most of the respective interval. To account for this and other data preprocessing challenges, a number of data preprocessing algorithms are implemented. *Figure **3*** to *Figure 7* show examples of data requested with the method *getValuesPeriodic(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 preprocessing with different algorithms.

In mode 1, a linear interpolation and temporally weighted arithmetic average is used for calculating periodic values. If the requested period contains more then 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 then 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 preprocessing algorithm and virtual datapoints, processing applications can query desired information based on appropriate software interfaces and data structures in real-time. This enables the processing application to focus on the respective use case.

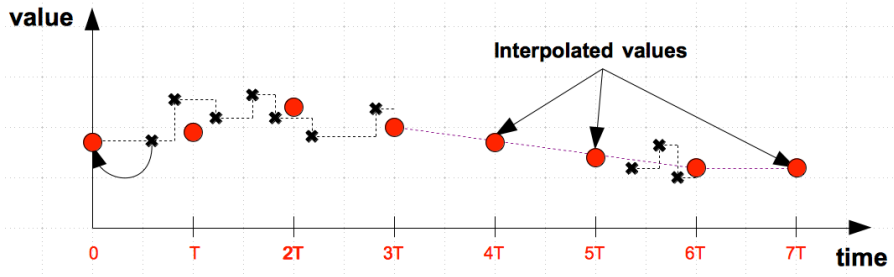

Figure 3. Data preprocessing - mode 1: time-weighted average / linear interpolation

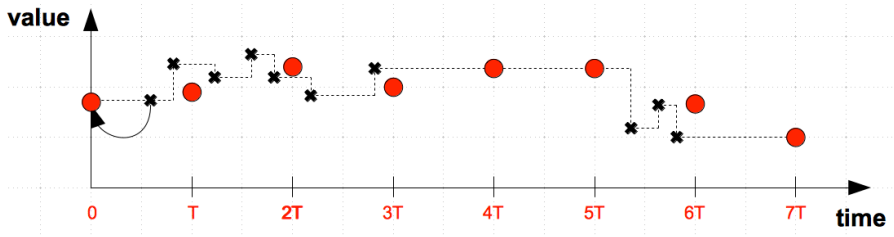

Figure 4. Data preprocessing - mode 2: time-weighted average / sample & hold

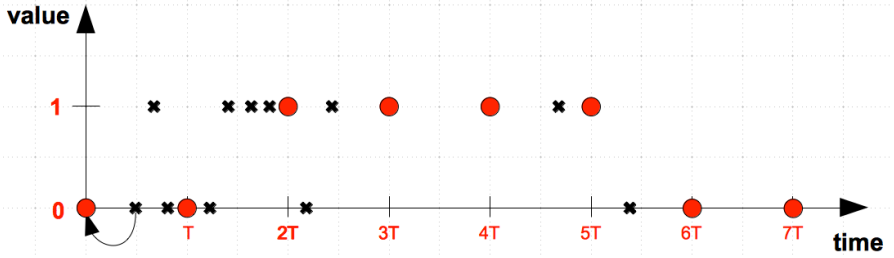

Figure 5. Data preprocessing - mode 3: majority decision / sample & hold

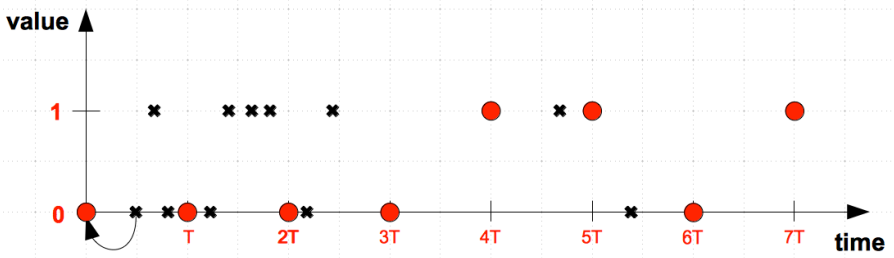

Figure 6. Data preprocessing - mode 4: forced 0 / default 1

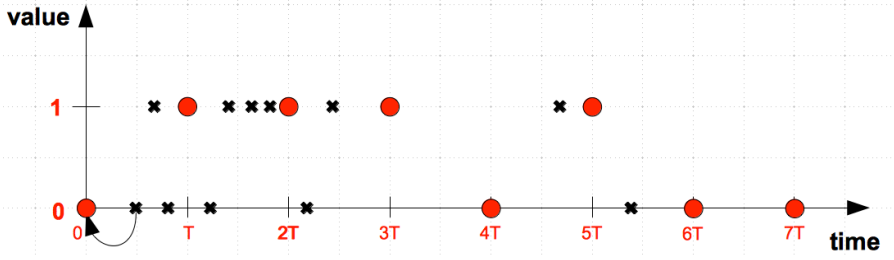

Figure 7. Data preprocessing - mode 5: forced 1 / default 0

STORAGE SERVICE

The storage service covers historical data access to measurements. Zach et al. 2012b shows how relational databases can be used to store desired data. Benchmark tests are used to examine the limits of this approach. Due to constraints (e.g., MySQL partitions can only split up values based on one criteria) for adequate separation of measurements in appropriate (independently indexed) dimensions, inefficient memory usage may result with relation databases. For example, in the proposed database design, all datapoints within one partition (timeframe) are cached (indexed) the same way, even if they are never used. Reduced memory usage could be achieved by keeping only the currently relevant datapoints and timeframes in memory. To overcome this limits, NoSQL big table technologies are promising. By improving the scalability of the storage service, additional data from the caching feature of virtual datapoints can be processed more efficiently.

The database schema discussed by Zach et al. 2012b introduces datapoints to store physical and virtual sensor data. Each datapoint is related to an arbitrary number of measurements (1-n relation). The measurements are separated into partitions to fit certain use cases. Performance optimization efforts primarily focus on the data entity, because most traffic deals with reading and writing sensor measurements. This raised the question if the sensor measurements can be stored and maintained separately to allow the appliance of scaling and performance strategies on sensor data without affecting the entire database schema. NoSQL solutions offer mechanisms that seem more suited to store sensor measurements than relational database systems. These include the ability to scale horizontally over multiple servers, a weaker concurrency model than ACID (Atomicity, Consistency, Isolation, Durability), efficient use of indexes and memory for data storage, and the possibility to dynamically add new attributes to data records (Catell 2010). Our building sensor data does not need to be type secure, because record security is not an issue. Therefore, changing the schema at runtime (e.g. adding new attributes to data tuples) offers possibilities to simplify the data handling. For example, the current implementation allows one timestamp by measurement. If a value does not change over time it is not possible to update the record without deleting the existing one. A NoSQL solution offers the possibility to add multiple timestamps to a measurement record. To reuse the processing algorithm developed in the current database design and to keep the benefits of a relational database management system for non-performance critical parts of the database schema, a mix of relation databases (MariaDB 2012) and NoSQL solutions (Cassandra 2012) is used. MariaDB is an open source replacement for MySQL that implements access to data stored in a NoSQL Cassandra cluster via the Cassandra Storage Engine (Cassandra SE, Figure 8).

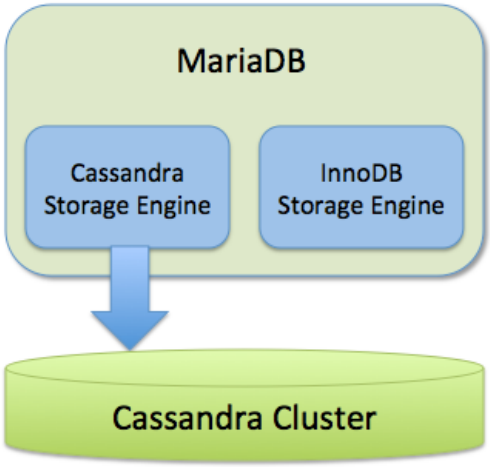

Figure 8: MariaDB/ Cassandra setup

Cassandra organizes data in column families that can be seen as a counterpart to tables in a relational database schema. Cassandra provides a simple data model that supports dynamic control over data layout and format. It was designed to handle high write throughput, while not sacrificing read efficiency (Lakshman and Malik 2010). The relational table that holds the sensor measurements maps to a Cassandra cluster. Instead of containing measurements, the relational table represents a view of a column family. The Cassandra SE defines an interface to query and manipulate (insert, update, delete) data in a cluster via common SQL commands.

The current implementation defines column families with a 36 sign long primary indexed key hash that references each row, a STRING referencing the datapoint name, a timestamp and a value record of type DOUBLE. The primary key determines which node the tuple is stored on. A secondary index is attached to the datapoint name column value to improve querying time by introducing a row cache. Whenever a certain datapoint name is called the entire record is pulled into memory and cached. Cache sizes are not set manually. Cassandra offers a possibility to weight cached data by access frequency and size automatically. The data is distributed across the nodes by using a MurMur3Partitioner. A 64-bit hashed value of the primary key (MurmurHash) is used to distribute the data evenly. The test wise implementation uses a replication factor of 1, which means that there is 1 copy of each data tuple on one node.

APPLICATION SZENARIOS

Virtual sensors can be used to obtain desired information based on real measurements. Due to various reasons pertaining to technical constraints (e.g., limits to the deployment of real sensors) and cost considerations, only a limited number of sensors are installed in real world buildings. The proposed framework enables so called virtual datapoints to include any kind of virtual sensor. By providing a uniform interface to real world and virtual sensors, client applications can process all building data streams consistently and focus on the respective use case. Complexity of integrated simulation tools is hidden from the processing application. Different building monitoring setups can be processed the same way.

If available sensors are not sufficient to calculate the requested information, additional devices can be added during the buildings lifecycle. Simulation powered virtual datapoints can use calibrated simulation models from the calibration service. If the accuracy of the simulation model is not sufficient, additional sensors can be installed to improve calibration. This approach allows collecting a rich data set with a minimum of installation costs. Virtual datapoints can also be used to calculate desired building performance indicators in real time. *Figure* ***9*** shows an overview of the proposed approach.

In many realistic circumstances, it is not possible to install monitoring systems with full building coverage. To address this issue, the simulation model of virtual datapoints can be calibrated based on monitored data obtained from a selected subset of building zones. Assuming that the monitored area has similar characteristics as the non-monitored area, virtual datapoints can be used for non-monitored areas based on the calibrated simulation model.

DISCUSSION AND CONCLUSION

The paper shows the usability of simulation-powered virtual sensors in a building information framework. A calibration service is used to provide accurate simulation models based on the recent history. Virtual sensors can derive information on parameters that are either difficult or expensive to measure. The relational database setup will be tested against the introduced MariaDB/Cassandra solution. A series of performance tests (write, read, concurrent write and read) should show if the hybrid database approach offers an alternative to the existing setup an can be used in a heavy-load production environment to provide efficient access to virtual datapoints.

ACKNOWLEDGEMENT

The research presented in this paper is supported by funds from the "Klima- und Energiefonds" within the program "Neue Energien 2020".

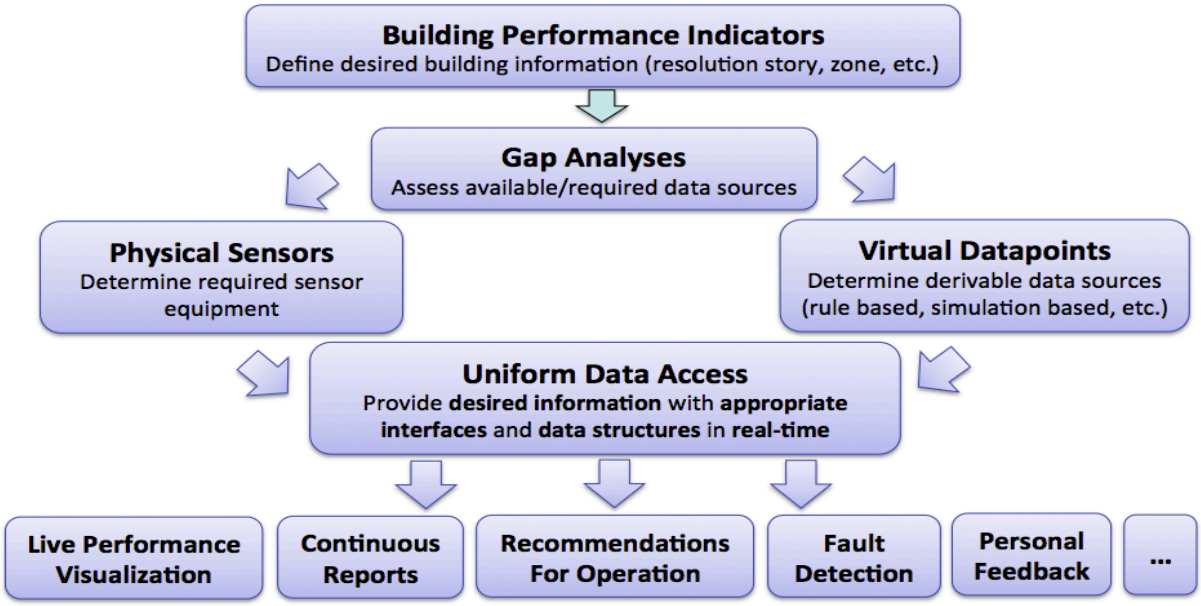

Figure 9. Building monitoring based on virtual datapoints

REFERENCES

Cassandra 2012. Big table NoSQL solution, January 2012, *http://cassandra.apache.org*

Cattell R., 2010. Scalable SQL and NoSQL data stores, ACM SIGMOD record Volume 39 Issue 4, Pages 12-27

DIN 1994. DIN EN 834, *Heat cost allocators for the determination of the consumption of room heating radiators - Appliances with electrical energy supply*

EnergyPlus 2012. EnergyPlus Energy Simulation Software, January 2012, *http://apps1.eere.energy.gov/buildings/energyplus/*

GWT-RPC 2012. Google Web Toolkit, January 2012, *http://code.google.com/webtoolkit/*

Java-RMI 2012. Java Remote Method Invocation, January 2012, *http://docs.oracle.com/javase/tutorial/rmi/*

Lakshman A, Malik P., 2010. *Cassandra: a decentralized structured storage system*, ACM SIGOPS Operating Systems Review Volume 44 Issue 2, Pages 35-40

MariaDB 2012. Open Source Database MariaDB, January 2012, *http://mariadb.org/*

MOST 2012. *Monitoring System Toolkit*, January 2012, http://most.bpi.tuwien.ac.at

MySQL 2012. Oracle MySQL, January 2012, *http://mysql.com*

OASIS 2012. *oBIX 1.0* Committee Specification 01, January 2012, *http://www.obix.org*

OPC UA 2012. OPC Foundation, January 2012, *http://www.opcfoundation.org*

Radiance 2012. Ray-tracing software, January 2012, *http://radsite.lbl.gov/radiance/*

Tahmasebi F. and Mahdavi A. 2012. *Monitoring-based optimization-assisted calibration of the thermal performance model of an office building*. Tirana, Albania.18.04.2012 - 21.04.2012. ISBN: 9789928-135-01-8; 5 S.

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

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

Zach R. 2012. *An open-source, vendor and technology independent toolkit for building monitoring, data preprocessing, and visualization.* Dissertation, Department of Building Physics and Building Ecology, 11.09.2012.