Generic Data Services for the Management of Large-scale Data Applications

Wolfgang Süß, Karl-Uwe Stucky, Wilfried Jakob, Heiko Maaß, Hüseyin K. Cakmak,
Institute for Applied Computer Science, Campus North
Karlsruhe Institute of Technology
P.O. Box 3640, 76021 Karlsruhe
Germany
wolfgang.suess@kit.edu http://www.iai.kit.edu

Abstract: A new metadata-driven concept of generic data management services for a wide range of applications for processing large-scale data of arbitrary structure is introduced. The novelty of the approach is a metadata structure that subdivides them into three basic metadata types according to their functionality: for the description of the structure and type of data, for the specification of key values for data identification, and for security and organizational purposes. A first implementation of core features and the feasibility of the concept is demonstrated by managing the data storage for an application, which measures and analyses voltage data from electrical power grids. Since electrical measurement data needs privacy protection, our proposal to address this requirement is outlined.

Key-Words: generic data services, metadata, object-oriented data management, large-scale data, data security

1 Introduction

The challenge of large-scale data management and analysis in different science communities leads to research and development in the areas of generic technologies and infrastructures for data management, access, and security. The SimLabs of KIT [1], community-specific Data Life Cycle Laboratories [2], and other users of the KIT Large-scale Data Facility (LSDF) [3,4] deal with large and differently structured data. Applications in major research fields like energy, climate, or particle physics range from modeling tools, simulation, and optimization to analyses of experimental data. Common demands are

- comfortable and reliable data storage and retrieval,
- a wide variety of search functions on different data structures, and
- high security and privacy levels.

The amount of data processed by the given applications varies from several gigabytes for object-oriented models up to hundreds of terabytes for experimental and simulation time series and their analysis.

In order to fulfill these requirements a generic data management system was introduced as a preliminary concept in [5] and [6]. Generic in this context denotes the independence from applications and storage systems or organizational types of storage like databases or file systems. The concept comprises the following features in addition to basic abilities, such as reading, writing, and deleting:

- Management of arbitrary structured data

Data objects consist of identifying data items, which must be readable independently of the platform, and optional arbitrarily structured user data. Data objects are returned in the same structure and granularity as they were stored.

- Identification of data objects by metadata

The identifying data items of an object are regarded as metadata which must be unique in their entirety. Such metadata can be used to search for single data objects or groups thereof.

- Security features, including
  - secure communication with the data management system,
  - safe storage, including measures against data loss,
  - data integrity and its verifiability,
  - accessibility restricted to authorized personnel, and
  - measures to ensure data privacy and compliance with data protection laws.
  - Integration of different types of storage systems.
  - This expanded view of the metadata concept confirms the change and development of their meaning and usage over time as described in detail by Mannens et al. [7], who point out an evolution of the role of metadata.

In this paper, we elaborate the concept of Generic Data Services (GDS), which is aimed at satisfying the above requirements based on object- and service-oriented principles. It will provide users with a convenient and secure way to store, manage, and retrieve data. Additionally, first prototype implementation will be reported.

The paper is organized as follows: After a discussion of related work in Section 2, we will introduce the overall concept of the GDS in detail in Section 3. This is followed by a description of the current implementation and first applications in Section 4. In Section 5 we summarize the paper and give an outlook on future work.

The heading of each section should be printed in small, 14pt, left justified, bold, Times New Roman. You must use numbers 1, 2, 3, … for the sections' numbering and not Latin numbering (I, II, III, …)

2 Related Work

Many Big Data studies point out that data management in science and technology today requires new methods, architectures, and software. The generic character of software modules and the key role of metadata are critical factors when coping with the new challenges. Following, a collection of references shall confirm this point of view.

Ailamaki et al. [8] examine techniques to address important requirements of general-purpose scientific data management. They identify scale, rate, and complexity as key issues that require new developments, including automation of metadata processing and data organization. Structured and unstructured data should be managed transparently; especially experimental data can benefit from object-oriented modeling. As soon as in 2008, Becla and Lim [9] in their first report on a workshop on extremely large databases outlined that assuming a perfect schema for storing structured and unstructured data seems to be inadequate.

Gray [10] suggested that it is necessary to support scientists by building generic tools and he gave a list of generic problems like e.g. a common schema, data organization, or building and executing models. It is this idea of a generic system for scientific and operational data management that motivated the developments presented in this paper.

As regards energy management systems, the IEC[1] 61970-403 [11] standard explicitly covers generic data access and shows the importance of transparent data management. It refers to data entities which are defined in the IEC 61970-3xx series as Common Information Model (CIM). These data are described in [12], where it is also stated that data services have to be "as generic as possible". IEC 61970-402 [13] standardizes common services for energy management systems and, inter alia, describes common resource identification and common resource description. In [14] the authors present a combination of CIM and OPC UA to build an ICT-architecture for the utility domain based on a semantic web services concept. OPC[2] Unified Architecture is a communication protocol that uses object-oriented techniques to generically handle typed instances [12,15].

Shahabi et al. [16] describe their OLAP[3] tool ProDA and show how database management systems have to be extended by analytical query capabilities to provide an appropriate data management for scientific analyses of large data sets. Szalay and Blakely [17] report on database-centric computing and emphasize the importance of building modular systems for large-scale data. Bender et al. [18] focus on the workflow properties of scientific data processing and analyzing. They propose a framework based on REST[4] services with a domain-independent generic architecture to fill the gap in their grid- and cloud-based systems concerning common data models and abstract interfaces. Another example is the SIMPL framework described in [19]. SIMPL extends the Business Process Execution Language BPEL by data management activities that provide abstract data access for scientific workflows. Parastatidis [20] suggests knowledge-oriented research infrastructures capable of linking information and semantics.

The key method to achieve flexibility of the desired generic data structures is their description by additional– metadata. Metadata are often described as data about data, which only shows that a lot of data can be considered metadata. It depends on the point of view which data are considered metadata in a given context. For this reason, different types of metadata may be defined depending on the kind of data the metadata are about. Marco [21], for example, considers metadata to be mainly historical data used in business processes. He distinguishes technical metadata and business metadata, the latter

[1] IEC: International Electrotechnical Commission (www.iec.ch)

[2] OPC: Object Linking and Embedding (OLE) for Process Control

[3] OLAP: Online Analytical Processing

[4] REST: Representational State Transfer

being of concern for application users. In [22] the authors mention provenance metadata as a special type of historical data in addition to metadata for certain combinations of metadata, and the implementation of metadata as objects themselves.

The publications mentioned here and many

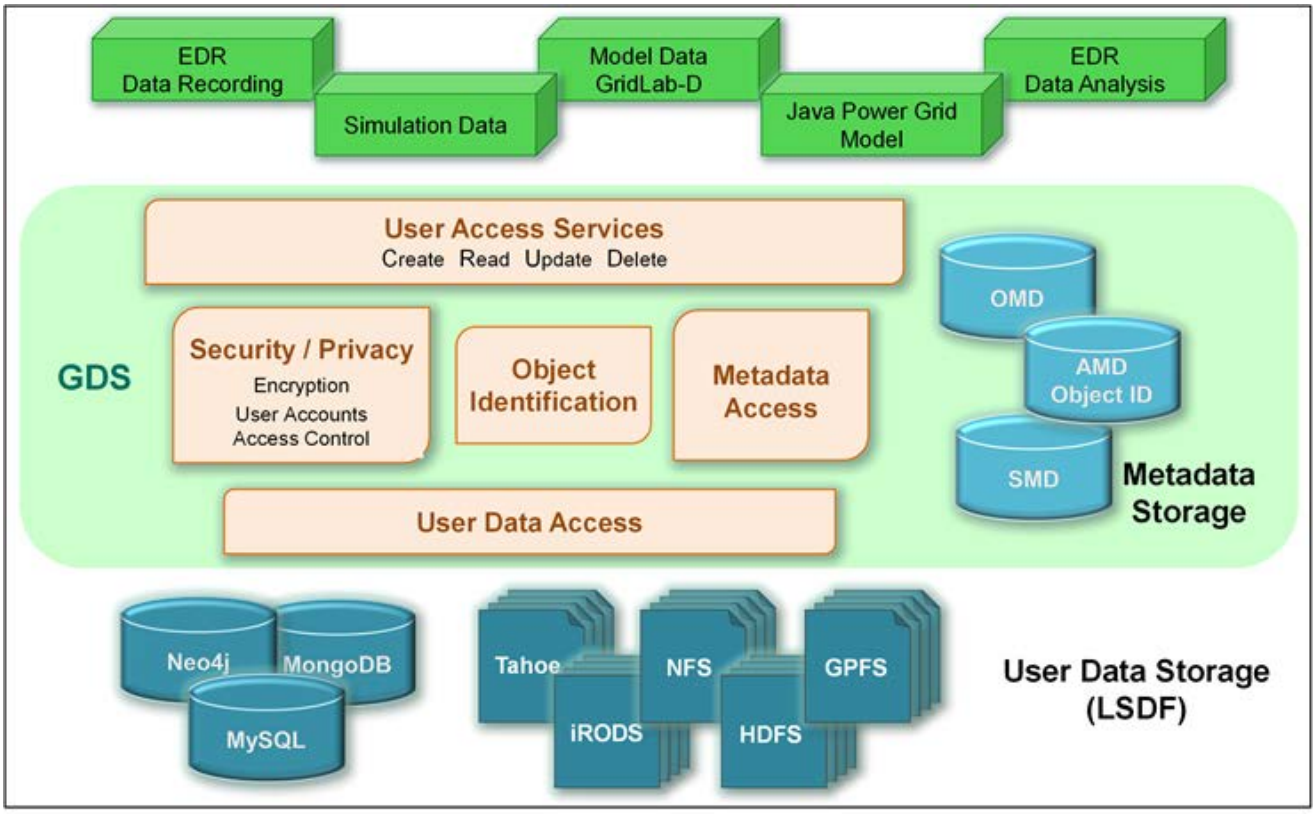

Fig. 1. GDS – Generic Data Services – currently comprise data access services to be used by applications, object identification, access to storage for user data and security services. It is based on three different types of metadata (see Sect. 3.1). The figure also shows applications and storage systems (databases and file systems) that are currently currently connected to GDS

storage and access and two user-centric types for user access arrangements and community-specific metadata. A prominent example of provenance metadata is the Dublin Core ([23] and [24]); it defines a set of elements that describe internet documents, which are frequently used in XML/RDF documents.

Kogalovsky [25] presented a paper with a systematic description of metadata as information resources. Important statements in this paper are the relative character of the separation into data and metadata, the characterization of descriptions of the data structure or format as metadata, and the distinction between application-dependent and –independent data.

The American National Information Standards Organization NISO [26] summarizes characterizations of metadata and different types of metadata, with structural metadata describing data objects, the embedding of metadata in objects or their separate storing, an identification possibility by others show that data management on large scales and for complex software and information systems will benefit from modularity, object- and service-oriented software designs, and generic data access concepts. It has to support distributed data storage and will need a well-structured metadata concept in particular.

3 Overall Concept of Generic Data Services

The objective of a generic data management system is the independence of the application, its data structures and data types, and of the underlying storage systems (Fig. 1). To achieve this, a semantically rich metadata concept has been developed and data are handled based on strictly object-oriented principles.

User data may but do not need to be structured in an object-oriented way. Therefore, it is referred to

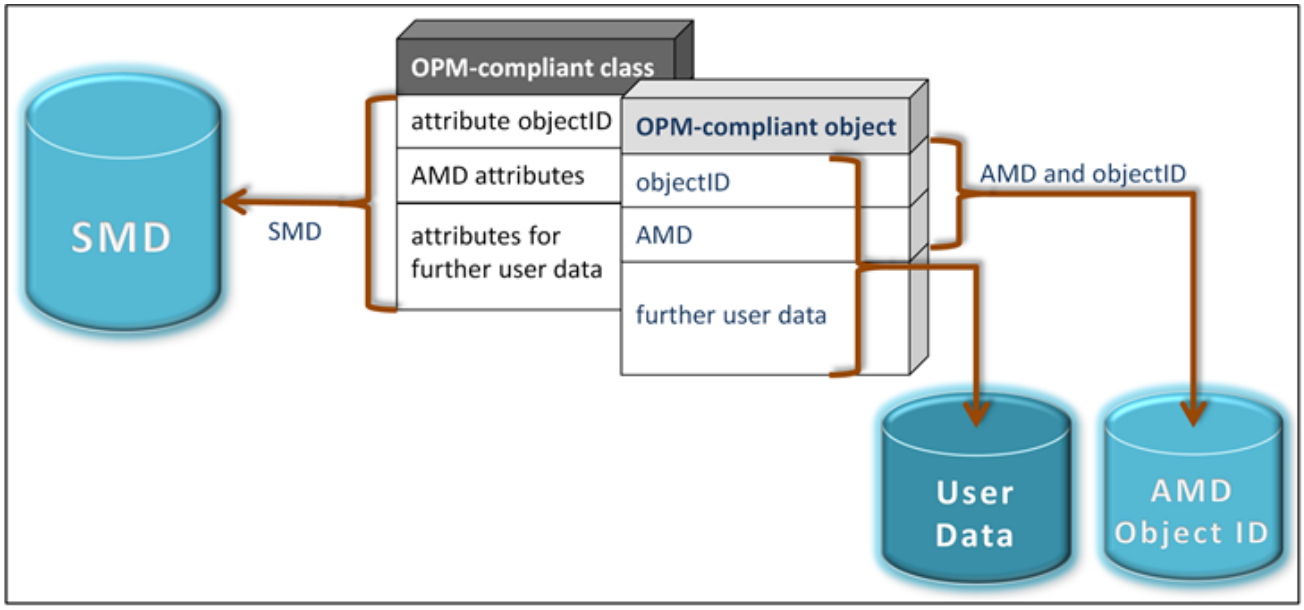

Fig. 2. The object-oriented programming model (OPM) defines a frame for the class and object structure. The explicit description of the classes is given by the structural metadata (SMD). An object used to store a user data element is stored in an appropriate storage device denoted by User Data. AMD and object IDs are stored

user *data elements* which themselves consist of *data items* of any cardinality that can be of standard data types like integer or string, or other data elements. Items containing unstructured data are treated as byte arrays. Data elements of the same structure and types are combined and the combination is called a *kind of user data* and given an identifying *name*. When looking at that in an object-oriented way, we have corresponding objects with their attributes and classes designated by their class names. Consequently, GDS distinguish between the world of user data and its GDS-internal object-oriented counterpart that will be created when GDS are configured for a certain application and kind of data (see Sect. 3.3).

3.1 GDS Metadata Types

GDS are based on metadata of three different types:

1. *Application metadata* (**AMD**): [25] explicitly mentions identification as a function that is supported by metadata. For the same purpose, GDS introduce AMD which comprise all data items of a user data element that are used by the application to identify a single data element. The AMD values serve as identifying key values of a data element and, thus, must be unique. Usually, they are not empty. AMD can be of any data type. The only situation, where no AMD are required, is described in the last paragraph of section 3.3.

2. *Structural metadata* (**SMD**): The concept of this type of metadata is of more general nature than the application presented here. It is aimed at describing an object-oriented class concept in general and the object-oriented programming model introduced in the next section in particular. In the GDS context, SMD are used to describe the classes corresponding to given kinds of user data. They are usually derived from a description of the data given by the user as discussed later in Sec 3.3.

3. *Organizational metadata* (OMD): They are used by the GDS to organize the localization of data elements in the storage system and to manage security features. The user is affected by the latter to some extent, as these data also include login data and access rights. They are described in Sect. 3.4.

AMD and OMD are described in more detail in the following sections starting with the object-oriented programming model, followed by the AMD and SMD.

3.2 Data Objects and Metadata

The GDS-internal representation of user data elements is objects that are compliant with an object-oriented programming model (OPM) which is the basis of all GDS software development. As shown in Fig. 2, an OPM-compliant class consists of three attribute sections. The first section holds an object identifier explained later, the second one consists of attributes containing the AMD, and the third one comprises further user data. The SMD

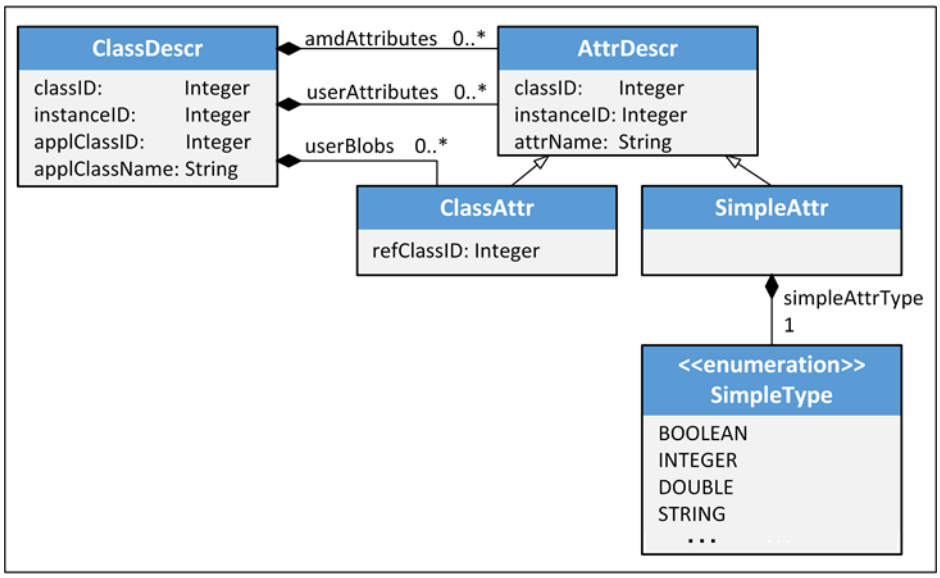

Fig. 3. Class schema of the **SMD**. SMD objects are used to describe a single user data element, which usually consists of AMD and may have further structured or unstructured user data items. Each structured data item can be either another object or of any simple data type like Boolean or Double

describe type and structure of the classes and of all attributes.

In the OPM all objects are identified by a system-wide object ID that meets all requirements for internal identification of an object, whereas AMD are an alternative means of identification mostly for humans and also applicable in software algorithms. In fact, each object ID consists of two IDs. The class ID identifies the object's class while the instance itself and is identified by the instance ID. Object IDs are provided and managed by a particular service so that their uniqueness is guaranteed. They are used for three purposes: To reference objects that are not stored in the main memory or remote objects, for administrative tasks like localization in GDS, and thirdly, the class ID is required for the SMD as described below. For both IDs eight-byte numbers are used by the current implementation, which is sufficient for about 3.4e+38 objects. For comparison, the estimated number of stars in the universe amounts to 10^{22} to 10^{24} [27].

As mentioned above, the SMD are used in GDS to describe the user data. For this purpose, user data elements of the same structure and item data types are treated equally and are represented by a class. Frequently, more or less large parts of the user data structure need not to be known by the GDS. In such cases, all or parts of these user data can be regarded and stored as unstructured data items comparable to binary large objects (blobs) in database applications. As shown in Fig. 2, the AMD and the object ID are stored separately as a metadata catalog by the GDS. Depending on their structure and user access requirements, the objects containing the complete user data are stored in appropriate storage devices for mass data. An application gets the data back, with their structure and granularity corresponding to those of the data passed to the GDS.

As SMD are an essential part of the GDS management functions and will be essential for all OPM-compliant software, they are explained in detail and by means of an example in the remainder of this section. The SMD consist of several classes. As they are OPM-conform, each data item has an object ID, i.e. a classID and an instanceID. To describe the class associated with a specific kind of user data, the SMD class ClassDescr is used, as shown in Fig. 3. Each ClassDescr consists of the attributes applClassID and a class name applClassName, which can be used by the application to address its specific kind of user data. The applClassID identifies the class that describes the kind of user data denoted by the corresponding applClassName. The class ClassDescr has three additional kinds of attributes, which describe the user data. The first category called amdAttributes is mandatory and describes the AMD. Its elements are AttrDescr objects, all of which have an attribute called attrName holding the name of the data item the AttrDescr object describes. The type of an AMD item can be either a simple data type like integer or string or it can be another object. In the latter case, the class of that object is described by a new SMD object, the applClassID of which is stored in the refClassID attribute. This implies a reference to the ClassDescr object of that AMD attribute. The userAttributes are based on the same AttrDescr class. The third category is called userBlobs and it is used to describe the optional unstructured user data, which are always stored as byte arrays. For consistency, this byte array is treated as any other

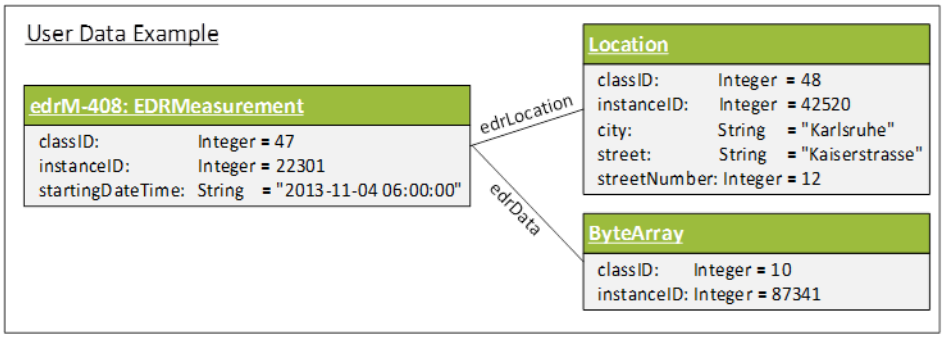

Fig. 4. Object diagram of a user data example consisting of one unstructured user data item, called edrData, and two AMD. One is a simple-type string holding the starting time of the measurement and the second one is an object called edrLocation of the class Location. The values of the classID attributes will be discussed in conjunction with Fig. 5

class and described by another ClassDescr object. The entire class structure is shown in the diagram of Fig. 3. It allows for full flexibility in describing the data items, including the AMD of user data elements, by SMD. From the point of view of the GDS, the SMD serve for administrative purposes of the management of user data and they are essential for a generic interface between the GDS and an application, as will be explained in more detail later on.

3.3 Use Case Electrical Data Recorder

The usage of the introduced SMD schema will be illustrated by a generalized example of user data. At this point, it is just of interest that they are generated by an electrical data recorder (EDR), which measures voltages in a power grid for a fixed time period and gives the results as time series, which are treated as unstructured data. Additionally, information about the location and the beginning of the measurement must be stored, which together are unique and, thus, serve as AMD. A corresponding OPM-compliant object diagram is shown in Fig. 4. Each data item is stored in an object of the class EDRMeasurement. Fig. 4 shows an object of an exemplary measurement, here denoted by edrM-408. This object contains two AMD items: The startingDateTime and the edrLocation, which itself consists of city, street, and streetNumber, thus resulting in four elementary AMD attributes. The measured time series data are stored in the edrData attribute. The meaning of the class IDs will be explained in the following discussion of the corresponding SMD. The instanceID values depend on the sequence of object creation and otherwise are arbitrary.

Fig. 5 shows how the three classes used to store the data of the given EDR example are described by SMD. The class EDRMeasurement is described by the object edrMeasurementClassDescr of the class ClassDescr, which contains the name of that user data class (here EDRMeasurement), its applClassID, and two attributes for the description of the user data. The first, amdAttributes, consists of two elements: One of class ClassAttr and one of class SimpleAttr. The latter describes the attribute startingDateTime of class EDRMeasurement. The other one describes the attribute edrLocation, which is an instance of the class Location to which the refClassID 48 refers. This class has three attributes of simple data types, as is shown by the object locationClassDescr in Fig. 5. They are described by the attribute attributes, which consists of three elements describing the two strings for city and street and the integer for streetNumber. Consequently, the edrLocation attribute of object edrM-408 in Fig. 4 holds the value 48 as classID. The only user data item in Fig. 4 that has not yet been covered is the unstructured edrData. It is described by the remaining userBlobs attribute of the SMD object edrMeasurementClassDescr. It is a ClassAttr with the attrName edrData and the refClassID 10. The refClassID refers to a class ByteArray. It is described by the SMD ClassDescr named byteArrayClassDescr, which has an applClassID of the same value. Again, this is the classID of edrData object of the user data example shown in Fig. 4.

3.4 Generic Application Interface

In general, user data can be of any structure. As a first step, we assume that the interface for exchanging data between an application and the GDS can be based on one of the widely spread

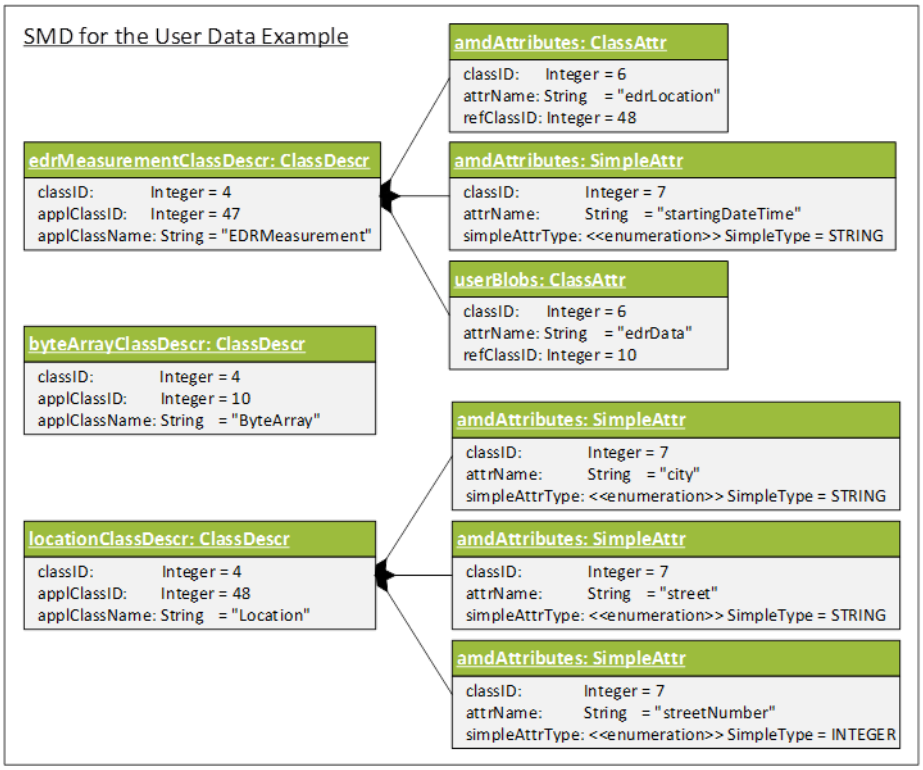

Fig. 5. Object diagram of the SMD for the EDR example of Fig. 4. Since the instance IDs are not needed in the present context, they are omitted here for reasons of clarity. Since classes like ClassDescr, ClassAttr, or SimpleAttr are used by the GDS in an early stage, they have low class IDs. Since the class ByteArray used for unstructured user data is something like a basic data type for the GDS, it is described by an SMD object, which is generated about as early as the standard classes for SMDs. Thus, it receives a very low applClassID

technologies for data description and exchange like XML or JSON.

Fig. 6 gives an overview of the generic data management system, which is divided into a part for data description (upper half) and one for data management (lower half). The integration of a new application starts with the definition of the structure of the user data and its basic types, which is supported by the User Data Description Editor UDDE. The editor guides the user in describing the data structures and assists with the definition of classes and attributes. The results are structural metadata (SMD) shown in the center of Fig. 6 as the linkage between the data description and data management parts of the GDS. The central role of the SMD is underlined in the figure by a lighter shade than the other data storage systems. Three components are based on SMD. First, the GDS needs SMD to manage the user data. Second, the class generator (CG) uses SMD to create OPM-compliant classes for the user data, which include methods for the construction of the objects and for their storage management. Third, the interface generator (IG) uses SMD to produce interface descriptions like XML or JSON schemata, which can be used by the user data interface of the application for serialization and deserialization of its data. A generated schema describes the data format of the interface between the user application and the GDS and is used by the GDS for data administration like writing, retrieving, or deleting user data. The GDS converts the data into OPM-compliant objects and stores them in the user data storage as shown in Fig. 6. Additionally, the AMD are stored separately, since they serve as a metadata catalog and are used to identify user data objects as described in the previous section.

For services like reading or deleting data, the objects must be addressed. This can be done by the application by directly providing AMD, including the name of that kind of user data, or by using object IDs In the latter case, the user data elements of the application have been generated as OPM objects or, in other words, we have an OPM-compliant application. Of course, such an application can alternatively use AMD.

3.5 Organizational Metadata and Security

Organizational metadata (OMD) serve three purposes:

- Localization of data objects and storage (management of storage access),
- management of security issues, and
- logging of data access, if required.

They will be described in more detail in the following sections.

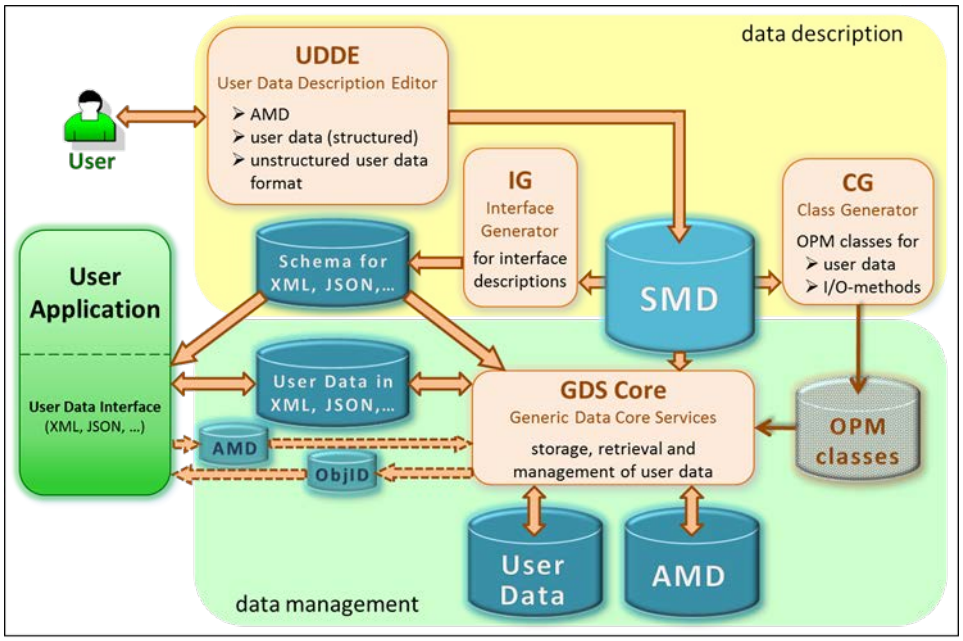

Fig. 6. Overall concept of the generic data management system with its central data storage system containing structural metadata (SMD). The figure shows data flow by big arrows and the generation of OPM-compliant classes in a suited programming language by small arrows.

3.5.1 Organizational Metadata for Object Localization

GDS comprise an OMD database that contains a table listing several OMD on a per-data-object basis (depicted in Fig. 7). The table provides several features. At first, it connects object ID with AMD (instanceID and AMD columns) and, thus, enables the GDS to switch between both options for object identification. The next part consists of the columns storageID and accessMD, which contain the localization data. storageID refers to the storage system, while the column accessMD contains information required for the access to the selected object inside this storage system, like the path and file name in case of a file system. GDS can use these localization data by connecting them with the identification information in the same table row.

The column objectSetID is relevant to security issues and points to an entry in another database table. The next section will go into details.

3.5.2 Management of Security Issues

Data objects shall be accessible by authorized personnel or authorized applications only. As it is expected that many objects can be treated equally with respect to access rights and ownerships, objects may be grouped into *object sets* and only these sets are subject to ownership and access rights. The aggregation of objects into sets is done according to rules given by the user based on application-dependent considerations. If single objects are to have their own rights, they must be placed in a one-membered object set. Object sets are identified and managed by an ID – the objectSetID in Fig. 7.

Furthermore, a user management is required. GDS users can own object sets and can be a member of one or more groups. Each object set is owned by exactly one user. There are three access rights (read, write, and delete). For an update operation, the combined rights of writing and deleting are required. As we only do data management, no execution rights are required. These three rights apply to owners of object sets and also to user groups to control the access of their members to object sets. Owner rights and the rights of one or more groups to access an object set are cumulative. As more than one group may have access to an object set, the approach is comparable to the concept of access control lists [28] in this respect. For the management of users, groups, object sets, and their access rights, another category of OMD is required. The concept of users, groups, and object sets is explained in more detail in [29].

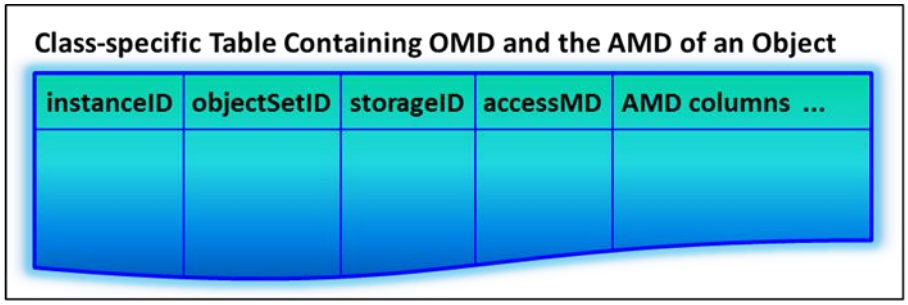

Fig. 7. Possible organization of AMD and the instanceID of an object in conjunction with OMD required for access (storageID, accessMD) and security (objectSetID)

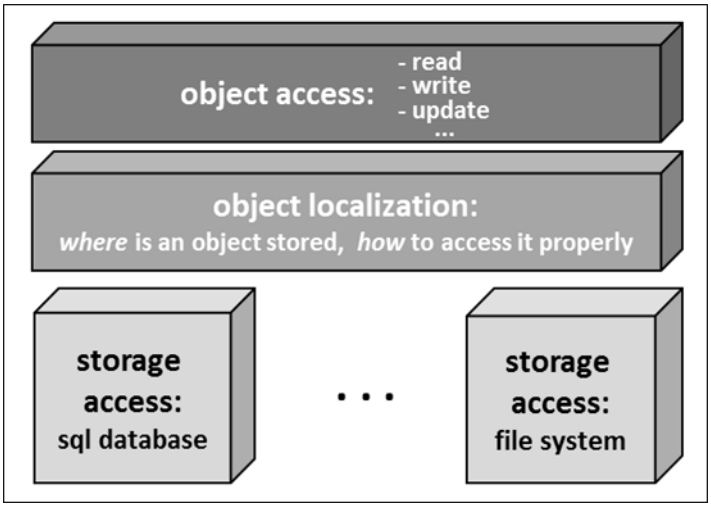

Fig. 8. Three layers of the generic interface to different storage systems. On the lowest layer, storage system-specific modules handle the details of the integrated storage systems and provide a neutral interface to the upper layer

To ensure data integrity, each stored data object is accompanied by a checksum to detect modifications. These checksums also belong to the OMD.

Logging of data access can be activated for object sets as a whole depending on the user and the type of access like reading or erasing. The information for this function is also part of the OMD.

Access to the GDS system is provided by a Virtual Private Network. The only exception is the local access of administrators to the servers. The mass storage system itself is accessed by secure protocols like sftp or ssh. Further details of the security concept can be found in [29].

3.6 Generic Storage Management

Storage systems currently in use or studied for usability within the scope of SimLab Energy [30,31] range from the file systems NFS [32], GPFS [33], Tahoe-LAFS [34], and HDFS [35] to the data system iRODS [36] and database systems MySQL [37], MongoDB [38], and Neo4j [39]. This single application scenario already demonstrates that the GDS are to be independent of the storage systems used. To achieve this, a hierarchy of three main layers is used, as shown in Fig. 8. The top layer offers storage-independent methods of generic objects for the basic access functions. Objects are identified by their object ID. The main function of the middle layer is the localization of an object: *Where* is the object stored and *how* can the storage system be accessed properly? This means access methods, authorization, file names, and the like. The user data serving as AMD will be stored in such a way that they can be used efficiently for search operations, while the complete object, which is expected to be much larger in most cases, may be stored in different storage systems, which are suited for handling large data objects. The details of the storage systems integrated are hidden to the middle layer by the lowest layer, which consists of storage system-specific modules. If necessary, packing and unpacking functions for large data systems can be located on this level.

In addition to the basic data access functions, the GDS offer an extended copy function, which copies data objects to a specified storage system to which the user application has access. The idea is to provide the user with his data or a subset thereof as required for a particular user application. These data objects are usually identified by their AMD, which can also be given as value ranges. This feature is required for data analysis of the SimLab Energy [40,41].

4 Implementation and First Applications

The first implementation of a basic version of the GDS is driven by mass data management for electrical data recorders and by power grid simulation data. This application is motivated by the present transition of the electrical supply system in Germany from a mainly centralized, nuclear or CO_2-intensive power generation with a hierarchical distribution network to a much more decentralized system and the utilization of renewable energies. To achieve a reliable control of such a power system based on much more volatile power generation, information about the exact system state, preferably in real time, is required [42]. On the other hand, detailed data on the system state are owned by the utilities or distributors and not comprehensively and freely available to researchers. Thus, information about the grid that is easily accessible and not limited in access by property issues is collected. As

energy feeding elements like photovoltaic farms connect to the low-voltage network and require advanced voltage control to maintain quality and stability [43], an investigation database is established by storing raw high-rate captures of the low-voltage network over a long term for large-scale comparison and later research.

As a consequence, a new electrical data recorder was developed [41], which produces a large amount of data, as will be described in the next section. The data measured among others serve as a basis for the verification of simulation models. Measured data, simulation models, and simulation results are managed by the GDS, as described in Section 4.2. Both applications, measurement and simulation, are briefly introduced here, mainly under the aspect of their data administration requirements, because their needs influence the implementation process to a great extent. For further details about the two applications and their motivation, the interested reader is referred to [44,45,41]. It should be pointed out here that the GDS are not limited to these applications, but that the underlying concept comprises all types of object-oriented data, as was described in the introduction.

4.1 Electrical Data Recorders

To record high-resolution time series of voltage and current measurements, the so-called "Electrical Data Recorder" EDR was developed. It consists of a 16-bit DAQ board, a GPS unit, and a computation unit with VPN-based internet access. Three input channels are used for the three-phase voltage measurement at a range between -430 V and +430 V with an accuracy better than 0.1%. Four more input channels can be used for synchronous current measurements. The typical rate is 12.8 kHz, while the adjustable rate is limited to 25 kHz [44]. One channel is used to simultaneously capture the GPS pulse-per-second signal for precise synchronization purposes. An EDR does some additional preprocessing and extracts features and characteristics of the measured time series according to EN50160 [46].

Based on the currently used rate of 12.8 kHz, an amount of 8.36 GiB per day and per EDR must be stored for three phase voltage captures. This is calculated as follows: Three channels times two bytes per sample times 12,800 per second times 4/3 for base-64 coding times 60 seconds, which, together with additional XML tags and preprocessed feature data, results in approximately 5.95 MiB per file and minute. Per day, 1440 of these files are generated, which together require 8.36 GiB of storage capacity [45].

EDR data are transferred as XML files and stored in objects of class EDRraw. They contain the time series data of a minute, which can be regarded as unstructured user data, and additional AMD. These AMD describe the pseudonymized measurement location and the recording time. For reasons of data protection, measurement locations are pseudonymized to EDRnnnn, where nnnn is a 4-digit number administrated by the EDR users of the GDS and not by the GDS itself. An example of an XML file is given in Figure 9. It shows the AMD of an object of class EDRraw and additional user data called data here, which serve as a filename. Note that the information about the class structure and the data types of the attributes is omitted in the XML file, as this information is part of the SMD of the class EDRraw.

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDRraw ...>
 <source value="EDR0003" />
 <year value="2013" />
 <month value="03" />
 <day value="15" />
 <hour value="11" />
 <minute value="53" />
 <data name="53.xml" />
</EDRraw>
```

Fig. 9. XML file of an EDR 1-minute measurement containing the AMD of the data set

Currently, EDR data are not returned individually for further processing, but copied in larger groups to a Hadoop file system. They are stored there using the file names which were given as additional user data when the object was stored. Selection is based either on one EDR device and location for a longer period of time or on several measurement sources and a short time range.

4.2 Power Grid Model Data

The medium-term goal of the *SimLab Energy* of KIT is the development of a simulation model of the power grid of the entire Campus North of KIT. For this purpose, the Electrical Grid Analysis Simulation Modeling and Visualization Tool *e*ASiMOV was developed, which integrates open source tools like GridLAB-D [41]. As it is to be as precise and close to reality as possible, the measured high-resolution data from the EDRs will be used to validate the model components and the model itself. First models based on GridLAB-D were developed

which can be managed and stored by the GDS. It is planned to perform simulation runs of larger parts of the KIT grid by *e*ASiMoV and to compare the resulting time series data with the data measured by the EDRs. The storage of the simulation results can also be managed by the GDS.

4.3 Secure Data Transfer

According to European and German law, measurement data of electrical power are subject to privacy protection, if the data allow conclusions to be drawn with respect to the personal power consumption and usage. This will be the case in most data acquisition situations. Consequently, the data must not only be stored and accessed securely as described in Sect. 3.4, but transport of the measured data must be protected against corruption and monitoring as well.

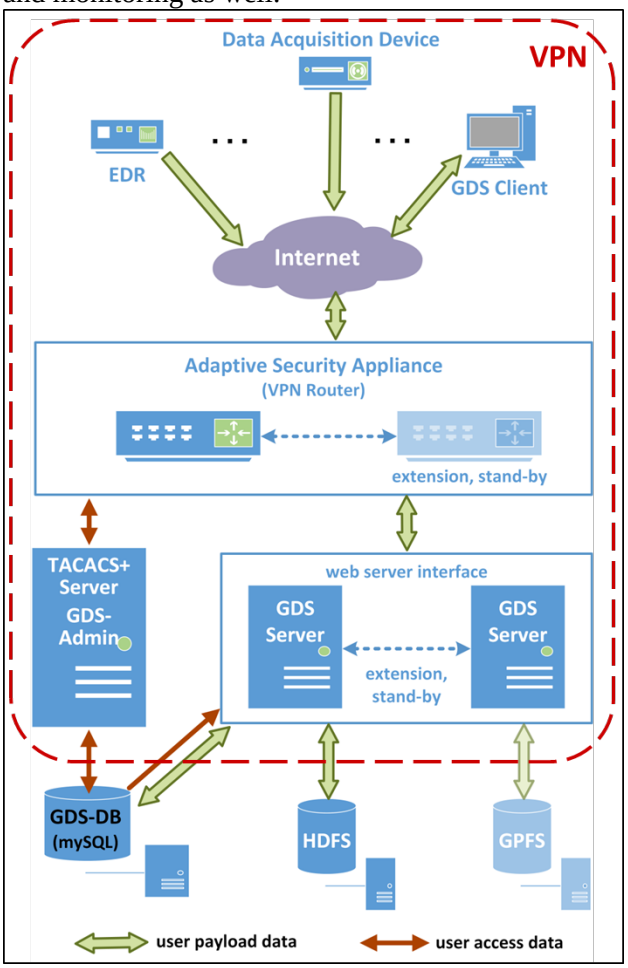

Fig. 10. Secure access to the currently implemented GDS via a VPN solution.

As described in detail in [29], this is achieved by a virtual private network (VPN), which is formed by the GDS server, the EDRs, other data acquisition devices, and GDS clients, see the upper part of Fig. 10. At present, one VPN router is active, while a second one is available as a stand-by device.

Only GDS users shall have access to the VPN. On the other hand, the whole system is part of the KIT network. Our solution is to have a GDS and VPN user administration, which is completely separated from KIT users. The GDS administration tool manages the users, groups of users, and their rights. It is based on a list of users allowed to access the GDS via VPN to the Tacacs+ server, which provides access management for the VPN router. Details of this highly scalable solution can be found in [29]. The GDS database server, which is visible within the local domain only, is accessible merely by its registered clients. At present, the HFDS server is being accessed via the KIT LAN, while the planned extension to the GPFS server will be done securely via ssh or scp.

4.4 Current State of Implementation

As stated before, the implementation of the concept is driven by the requirements of first applications and its users. At present, a basic version of the GDS is implemented, which covers a part of the components shown in Fig. 6. The core functionality of the GDS comprises storage of objects and retrieval via AMD. The implementation of user- or group-based access rights is under way. First versions of the UDDE and of the IG are implemented. The database scheme for SMD has been developed and a MySQL database for SMD storage has been set up and is about to be adapted to the current GDS. Various solutions for OPM object serialization are being tested for applicability within the interface generator. These new components will allow for a substitution of the currently handmade SMDs and XML schemata by automatically generated ones.

The GDS server is already working in a hot standby and a simple load balancing policy: Five servers are working redundantly in parallel. At present, the load is small compared to the capacity of one server. This mode of operation is mainly aimed at offering a highly available service based on comparably less reliable standard hardware.

5 Summary and Outlook

We introduced our new approach of metadata-driven generic data management services (GDS) and presented the underlying concept of application, structural, and organizational metadata. This approach allows for flexible management of

different data structures based on an object-oriented view. The current version of structural metadata (SMD) has been highlighted in detail, since SMD are essential for the object-oriented handling of application data. A first implementation for the management of electrical measurement data was described, with emphasis placed on associated security issues due to their importance to energy data life cycles. The use of standard techniques for secure data transmission has the advantage of relying on the services of commercial providers for future innovations and challenges.

Future work will concentrate on further developments of all GDS components, in particular of the user data description editor UDDE and of the interface generator. The latter shall serialize user data elements for data transfer to the GDS and GDS data objects vice versa. The concept of structural metadata will be enhanced so that it can deal with hierarchical nets of objects and classes, as it will be required for a detailed storage of model data of e.g. power grids. This development will be guided by the idea of ontology-based database access [47]. The frequent repetition of metainformation makes XML a memory-intensive exchange format. This clear drawback is reduced to some extent by JSON, but not solved really. Thus, we intend to develop a more compact and much less redundant exchange format based on SMD.

The integration of SMD into the core of GDS will progress into several directions: A class generator that produces Java classes has to be established. SMD can be a basis of generically invoking user interfaces for administrative access to the GDS, for front ends that have to be tailored to user wishes or even for generating the user interfaces of UDDE. SMD can also be utilized for generically storing data in various services of the third storage system-specific layer in Fig. 8.

The strict requirements relating to security and privacy in energy applications will continue to improve the standard method-based solutions, mainly in the fields of anonymization, pseudonymization, and encryption.

References:

[1] Kirner, O. KIT - SCC - Research and Innovation - Computational Science and Engineering - Simulation Laboratories. http://www.scc.kit.edu/en/research/5960.php (accessed on 22 February 2016).

[2] Meyer, J. LSDMA - The Project. http://www.helmholtz-lsdma.de/56.php (accessed on 22 February 2016).

[3] García, A.; Bourov, S.; Hammad, A.; van Wezel, J.; Neumair, B.; Streit, A.; Hartmann, V.; Jejkal, T.; Neuberger, P.; Stotzka, R. The Large Scale Data Facility: Data Intensive Computing for Scientific Experiments. In 25th IEEE Int. Symp. on Parallel and Distrib. Processing, IPDPS 2011, Workshop Proc., Anchorage, Alaska, USA, 16-20 May 2011; IEEE: Piscataway, NJ, 2011; pp 1467–1474.

[4] van Wezel, J. KIT - SCC - Research and Innovation - Data Management, Data Analysis and secure IT Federations - Large Scale Data Management & Analysis. http://www.scc.kit.edu/en/research/lsdf.php (accessed on 22 February 2016).

[5] Maaß, H.; Çakmak, H.K.; Süß, W.; Quinte, A.; Jakob, W.; Muller, E.; Boehm, K.; Stucky, K.-U.; Kuehnapfel, U.G. Introducing a new voltage time series approach for electrical power grid analysis. In 2012 IEEE Int. Energy Conf. and Exhib. (ENERGYCON), Florence, Italy, 9-12 Sept. 2012; pp 890–895.

[6] Maaß, H.; Çakmak, H.K.; Süß, W.; Quinte, A.; Jakob, W.; Stucky, K.-U.; Kuehnapfel, U.G. Introducing the Electrical Data Recorder as a new capturing device for power grid analysis. In IEEE Int. Workshop on Applied Meas. for Power Systems (AMPS 2012), RWTH Aachen, Germany, 26 - 28 Sep 2012; pp 1–6.

[7] Mannens, E.; Verborgh, R.; van Hooland, S.; Hauttekeete, L.; Evens, T.; Coppens, S.; van de Walle, R. On the Origin of Metadata. Information 2012, 3, 790–808, doi:10.3390/info3040790.

[8] Ailamaki, A.; Kantere, V.; Dash, D. Managing scientific data. Commun. ACM 2010, 53, 68–78, doi:10.1145/1743546.1743568.

[9] Becla, J.; Lim, K.-T. Report from the First Workshop on Extremely Large Databases. Data Sci. J. 2008, 7, 1–13, doi:10.2481/dsj.7.1.

[10] Hey, T.; Tansley, S.; Tolle, K. Jim Gray on eScience: A Transformed Scientific Method. In The Fourth Paradigm: Data-Intensive Scientific Discovery, Second printing, Version 1.1; Hey, T., Tansley, S., Tolle, K.,

Eds.: Redmond, Washington, 2009; pp xvii–xxxi.

[11] International Electrotechnical Commission IEC. Energy management system application program interface (EMS-API) – part 403: Generic Data Access, 2009 (IEC 61970-403).

[12] Uslar, M.; Specht, M.; Rohjans, S.; Trefke, J.; González, J. The Common Information Model CIM. IEC 61968/61970 and 62325 -- a practical introduction to the CIM; Springer: Berlin, New York, 2012.

[13] International Electrotechnical Commission IEC. Energy management system application program interface (EMS-API) – part 402: Common Services, 2009 (IEC 61970-402).

[14] Rohjans, S.; Uslar, M.; Juergen Appelrath, H. OPC UA and CIM: Semantics for the smart grid. In Proc. of Transm. and Distrib. Conf. and Exposition (T&D), 2010 IEEE PES, New Orleans, LA, USA, 19 - 22 Apr 2010; pp 1–8.

[15] Mahnke, W.; Leitner, S.-H.; Damm, M. OPC Unified Architecture; Springer: Berlin Heidelberg, 2009.

[16] Shahabi, C.; Jahangiri, M.; Banaei-Kashani, F. ProDA: An End-to-End Wavelet-Based OLAP System for Massive Datasets. Computer 2008, 41, 69–77, doi:10.1109/MC.2008.130.

[17] Szalay, A.; Blakeley, J. Gray's Laws: Database-centric Computing in science. In The Fourth Paradigm: Data-Intensive Scientific Discovery, Second printing, Version 1.1; Hey, T., Tansley, S., Tolle, K., Eds.: Redmond, Washington, 2009; pp 5–11.

[18] Bender, A.; Poschlad, A.; Bozic, S.; Kondov, I. A Service-oriented Framework for Integration of Domain-specific Data Models in Scientific Workflows. Procedia Comput. Sci. 2013, 18, 1087–1096, doi:10.1016/j.procs.2013.05.274.

[19] Reimann, P.; Reiter, M.; Schwarz, H.; Karastoyanova, D.; Leymann, F. SIMPL - A Framework for Accessing External Data in Simulation Workflows. In BTW 2011, Datenbanksysteme für Business, Technologie und Web ; 14. Fachtagung GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Kaiserslautern, Germany, 02. - 04.03.2011; Härder, T., Ed.; Ges. für Informatik: Bonn, 2011; pp 534–553.

[20] Parastatidis, S. A Platform for All That We Know: Creating a Knowledge-Driven Research Infrastructure. In The Fourth Paradigm: Data-Intensive Scientific Discovery, Second printing, Version 1.1; Hey, T., Tansley, S., Tolle, K., Eds.: Redmond, Washington, 2009; pp 165–172.

[21] Marco, D. Building and managing the meta data repository. A full lifecycle guide; Wiley: New York, 2000.

[22] Hey, T.; Trefethen, A. The data deluge: an e-Science perspective. In Grid computing: Making the global infrastructure a reality; Berman, F., Fox, G., Hey, A.J.G., Eds.; Wiley: Chichester, England, New York, 2003; pp 809–824.

[23] DCMI. DCMI Home: Dublin Core® Metadata Initiative (DCMI). http://dublincore.org/ (accessed on 22 February 2016).

[24] Marco, D.; Jennings, M. Universal meta data models; Wiley: Indianapolis, Ind, 2004.

[25] Kogalovsky, M.R. Metadata in computer systems. Program. Comput. Soft. 2013, 39, 182–193, doi:10.1134/S0361768813040038.

[26] National Information Standards Organization NISO. Understanding Metadata. http://www.niso.org/publications/press/UnderstandingMetadata.pdf (accessed on 22 February 2016).

[27] esa. How many stars are there in the Universe? / Herschel / Space Science / Our Activities / ESA. http://www.esa.int/Our_Activities/Space_Science/Herschel/How_many_stars_are_there_in_the_Universe (accessed on 22 February 2016).

[28] IETF / ISOC. RFC 4949 - Internet Security Glossary, Version 2. https://tools.ietf.org/html/rfc4949 (accessed on 22 February 2016).

[29] Kramer, A.; Jakob, W.; Maaß, H.; Süß, W. Security in Large-scale Data Management and Distributed Data Acquisition. In Proc. of the 3rd Int. Conf. on Data Manag. Technol. and Appl. (DATA 2014), Vienna, Austria, 29 – 31 August 2014; Helfert, M., Holzinger, A., Belo, O., Francalanci, C., Eds.; INSTICC, Portugal, 2014; pp 125–132.

[30] Süß, W.; Stucky, K.-U.; Quinte, A.; Jakob, W. Performance Tests of Energy Data Storage Using Different Distributed Parallel File Systems. In Latest trends in applied informatics and computing, Proc. of the 3rd Int. Conf. on Appl. Inform. and Comput. Theory (AICT'12), Barcelona, Spain, October

17-19, 2012; Carstea, C.-G., Ed., 2012; pp 123–128.

[31] Koch, D.P. KIT - SCC - Research and Innovation - Computational Science and Engineering - Simulation Laboratories - SimLab Energy. http://www.scc.kit.edu/en/research/8037.php (accessed on 22 February 2016).

[32] Sandberg, R. The Sun Network File System: Design, Implementation and Experience. In Proc. of the USENIX 1986 Summer Technical Conf. and Exhib., 1986. http://www.cs.swarthmore.edu/~newhall/readings/nfs.pdf (accessed on 22 Febr. 2016).

[33] Schmuck, F.; Haskin, R. GPFS: A Shared-Disk File System for Large Computing Clusters. In Proc. of the 1st USENIX Conf. on File and Storage Technol. (FAST 2002), Monterey, CA, USA, January 28-30, 2002.

[34] Wilcox-O'Hearn, Z.; Warner, B. Tahoe: The Least-authority Filesystem. In Proc. of the 4th ACM Int. Workshop on Storage Security and Survivability, Alexandria, VA, USA, October 27 - 31, 2008; ACM: New York, NY, 2008; pp 21–26.

[35] Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In MSST '10 Proc. of the 2010 IEEE 26th Symp. on Mass Storage Systems and Technol. (MSST), Lake Tahoe, NV, USA, 6 – 7 May 2010; pp 1–10.

[36] Hünich, D.; Müller-Pfefferkorn, R. Managing Large Datasets with iRODS - a Performance Analysis. In Proc. of the 2010 Int. Multiconf. on Comput. Sci. and Inf. Technol. (IMCSIT), Wisla, Poland, 18 - 20 Oct. 2010; Ganzha, M., Ed.; IEEE: Piscataway, NJ, 2010; pp 647–654.

[37] Pachev, S. Understanding MySQL internals, 1st ed.; O'Reilly: Beijing, Sebastopol, CA, 2007.

[38] Redmond, E.; Wilson, J.R.; Carter, J. Seven databases in seven weeks. A guide to modern databases and the NoSQL movement; Pragmatic Bookshelf: Dallas, Texas, 2012.

[39] Jordan, G. Practical Neo4j; APress: Berkeley, CA, 2014.

[40] Bach, F.; Çakmak, H.K.; Maaß, H.; Kuehnapfel, U. Power Grid Time Series Data Analysis with Pig on a Hadoop Cluster Compared to Multi Core Systems. In Proceedings of the 2013 21st Euromicro Int.Conf. on Parallel, Distributed, and Network-based Processing, Belfast, UK, 27 February-1 March 2013; Stotzka, R., Milligan, P., Kilpatrick, P., Eds.; IEEE Computer Society; IEEE: Los Alamitos, Calif, Piscataway, N.J, 2012; pp 208–212.

[41] Maaß, H.; Çakmak, H.K.; Süß, W.; Quinte, A.; Jakob, W.; Stucky, K.-U.; Kuehnapfel, U.G. First Evaluation Results Using the New Electrical Data Recorder for Power Grid Analysis. IEEE Trans. Instrum. Meas. 2013, 62, 2384–2390, doi:10.1109/TIM.2013.2270923.

[42] Bakken, D.; Bose, A.; Hauser, C.; Whitehead, D.; Zweigle, G. Smart Generation and Transmission With Coherent, Real-Time Data. Proceedings of the IEEE 2011, 99, 928–951, doi:10.1109/JPROC.2011.2116110.

[43] Stetz, T.; Yan, W.; Braun, M. Voltage Control in Distribution Systems with High Level PV-Penetration. In Conf. Proc. of the 25th Eur. Photovolt. Solar Energy Conf. and Exhib., Valencia, Spain, 6 - 10 Sept 2010; WIP-Renewable Energies: München, 2010; pp 5000–5006.

[44] Maaß, H.; Çakmak, H.K.; Bach, F.; Kühnapfel, U. Preparing the Electrical Data Recorder for Comparative Power Network Measurements. In 2014 IEEE Int. Energy Conf. and Exhib. (ENERGYCON), Dubrovnik, Croatia, 13-16 May 2014, 2014; pp 759–765.

[45] Maaß, H.; Cakmak, H.K.; Bach, F.; Mikut, R.; Harrabi, A.; Süß, W.; Jakob, W.; Stucky, K.-U.; Kühnapfel, U.G.; Hagenmeyer, V. Data processing of high-rate low-voltage distribution grid recordings for smart grid monitoring and analysis. EURASIP J. Adv. Signal Process. 2015, 2015, 47, doi:10.1186/s13634-015-0203-4.

[46] DIN. Voltage characteristics of electricity supplied by public distribution networks; Beuth Verlag: Berlin, 2016, EN 50160:A1. http://www.din.de/en/getting-involved/standards-committees/dke/standards/65212!search-na?query=EN+50160 (accessed on 22 February 2016).

[47] Kogalovsky, M.R. Ontology-based data access systems. Program. Comput. Soft. 2012, 38, 167–182, doi:10.1134/S0361768812040032.