View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 1087 – 1096

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

A service-oriented framework for integration of domain-specific data models in scientific workflows

Andreas Bender[a], Angela Poschlad[a], Stefan Bozic[a], Ivan Kondov[a,*]

[a]*Steinbuch Centre for Computing, Karlsruhe Institute of Technology (KIT), P.O.Box 3640 76021 Karlsruhe, Germany*

Abstract

Managing data exchange in scientific workflow simulations is a challenge for which existing solutions have only limited success. Basing on a data-model approach we developed a service-oriented, platform- and language-independent framework which provides seamless access for heterogeneous applications to data storage resources distributed over grids and clouds. After implementing a working prototype of the framework we demonstrated its benefits for a complex workflow application from the domain of multiscale materials modeling. Because of its generic architecture and wide standards conformity the framework can be deployed as grid or cloud middleware for various other domains of computational science to enable rapid development of complex workflow applications.

Keywords: multiscale materials modeling; simulation workflow; simulation dataflow; data model; service-oriented architecture; web services

1. Introduction

Computer simulation is very important for research in scientific fields such as materials science and nanotechnology. Substantial efforts, both in industrial and in publicly funded research, have been made in modeling and simulation of whole devices, e.g. organic light emitting diodes. Such simulation is very demanding because of the unfavorable computational scaling with material size and requires modeling on several length scales from the nanometer up to the micrometer range. Therefore, multiscale modeling in materials and nanomaterials sciences [1, 2] is a promising paradigm that makes even very large and complex problems physically tractable and the computer simulation feasible. Using sequential multiscale modeling [3] a computer simulation includes several sequential or parallel programs which are loosely coupled and do not contain cross-dependencies. Thereby, the so-called workflow formally describes the process of execution and the executed programs are often called workflow steps.

However, multiscale modeling implies high complexity caused by the interaction of several individual physical models on different scales and the large number of model parameters. Multiscale models contain program codes from very different domains of expertise and cannot be employed in a straightforward way by end-users, e.g. industrial researchers or experimentalists from academia. On the other hand, current environments for modeling and simulation provided at computing centers are too complicated for non-experts. Since computer simulation

*Corresponding author. Tel.: +49-721-608-28644; Fax: +49-721-608-24972.
E-mail address: ivan.kondov@kit.edu.

1877-0509 © 2013 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-ND license.
Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.274

is typically planned and carried out by scientists who are non-experts in the technical fields of high performance computing and computer science, a corresponding easy-to-handle software infrastructure must be provided.

A further challenge with sequential modeling is the increasing number of different programs that a group or an individual researcher have to learn and use quickly. The programs must be linked to work together with a lot of effort. The scientist designing applications with multiscale models has to cope with rapidly increasing inventory of multiple different programs which carry out very often the same functions. For instance, if one function has to be changed the scientist has to change all programs with a lot of effort. Efficient and sustainable solutions have been sought in the currently running projects MAPPER (www.mapper-project.eu) and MMM@HPC (www.multiscale-modelling.eu). An integration of components adopting the paradigm of service-oriented architectures (SOA) has been recently proposed as an effective solution and demonstrated to work [4, 5, 6].

Nevertheless, the major challenge is the handling of data exchange between the workflow steps which is often referred to as *dataflow*. The data, which is fairly complex for multiscale materials modeling, must be stored in such a way that all workflow steps can easily access them. However, different program codes in the most cases cannot use the same data source because they have mutually incompatible data representations, heterogeneous data formats or non-uniform data access interfaces. The data must therefore be converted between the workflow steps into a variety of supported formats. This strategy requires continuous and laborious re-implementation of multiple converter programs for every new combination of simulation codes in a workflow application [7]. Moreover, multiple conversions of large data can become a bottleneck and even make the workflow simulation unfeasible.

With the present work we provide a generic solution for modeling and management of data in scientific workflows in a simple and uniform way. We report on a further development of the concept from [7] which has now flowed into a novel framework with service-oriented architecture based on modern standards for web services. In addition, we have implemented a prototype of the framework to demonstrate its functionality and general applicability for a multiscale modeling application in computational science. The basic concepts adopted here have been well established in the computer science, however, insufficiently used for multiscale modeling. Our practical experience in computational materials science projects has shown that domain experts are unaware of or not able to use these technologies because of the knowledge gap and lack of ready-to-go tools on the level of IT expertise of a domain expert. With this contribution we make these concepts readily accessible and usable for multiscale modeling where they are most beneficial.

The paper is organized as follows. In the next Section 2 we describe our previous work and review other approaches that address some aspects of the dataflow challenge. The design of the framework and its implementation are outlined in Sections 3 and 4, respectively. Sections 5 and 6 illustrate the usage of the framework implementation pointing out major advantages comparing to previously employed techniques. In Section 7 we summarize the key results of this work and suggest directions for future work.

2. State of the art

The GridBean Application Programming Interface (API) [8] and the UNICORE workflow engine [9] provide very flexible infrastructure for application integration in workflows, facilitating the management of storage and staging of input and output data. In particular, input and output files of GridBeans connected in a workflow can be shared using a special UNICORE workflow storage. The OpenMolGRID project [10, 11] has developed an open source software infrastructure for in-silico drug design on the Grid and provided solutions for data warehousing of molecular structures [11, 12], data mining, molecular engineering, grid integration, and process automation. In previous work, we have adopted the UNICORE middleware [13], GridBeans and the OpenMolGRID library to construct complex workflow applications for multiscale materials modeling and demonstrated their operativeness and performance in a proof-of-principle simulation of organic light emitting diodes [4, 5].

However, we have learned that integration of a common data model and abstracted interfaces between application steps and distributed storage are still necessary to efficiently tackle the dataflow. Unfortunately, neither UNICORE middleware nor OpenMolGRID library provide means for handling the inherent structure of input and output data. Also the common approach to write “all-to-all” format converters does not scale well. In particular, we realized that the workflow designer needs a common data model which allows different workflow components to share data. Moreover, the data should be accessible from each individual application code via an interface in

the programming language of the relevant application or, even better, through a language-independent interface (e.g. a web service) to allow rapid implementation of pre- and post-processors and construction of workflows from standard components.

In more recent work, considerable effort has been spent for reorganization of the package structure of the OpenMolGRID library [7]. For example in the *Format* package new classes have been written for interfacing data of individual workflow steps. Also classes in the *Process* package have been elaborated providing the ability to monitor the I/O streams of the wrapped application processes. This work was a considerable progress towards a full-featured dataflow management system, however, the main aspects of abstraction from storage type and format as well as handling of data models have stayed open so far.

In the following, we give a short overview of other approaches aiming at solving the dataflow challenge. Recently the SIMPL framework architecture has been proposed for access to heterogeneous data sources in scientific workflows [14]. The SIMPL framework has been designed as an extension to existing scientific workflow management systems to provide abstraction for data management and data provisioning in scientific and engineering simulation workflows. However, the SIMPL framework does not provide meta-models or other means for data modeling.

In the field of chemistry the Chemical Markup Language (CML) [15] provides a solution for data representation, storage and exchange. Recently, the CML content model has been loosened giving more freedom to scientists of different domains of chemistry to represent their data in a flexible, understandable and still machine-readable way, i.e. conforming to the CML schema, achieved by adopting the concept of dictionaries and conventions [16].

Attempts to develop a generic methodology for integration of data and algorithms into composite applications have been done using the concept of ontologies. Ontologies have been supported in Chemomentum data services [17] for molecular engineering and employed for multiscale modeling in chemical engineering [18, 19]. In the framework of the Integrated Tokamak Modeling Task Force (ITM), the Universal Access Layer (UAL) [20] has been developed to provide capability of storing and retrieving data in simulations using the Kepler workflow system. The underlying hierarchical data structure is based on the storage formats MDSplus and HDF5 and the granularity in data access is defined by a set of so-called Consistent Physical Objects with the use of ontologies [21].

A concept for data interchange for computational NMR has been employed in the Collaborative Computational Project for NMR (CCPN) [22] including a generic data model and API code generation engine [23]. The system is restricted to a few languages (Python, C and Java) and storage types (XML files and SQL databases). The developed software suite CcpNmr has been provided publicly as open source and used recently at our site to integrate a program for computational NMR [24]. In another approach for biomolecular NMR data analysis, workflow models and conceptual and logical data models [25] have been proposed which have led recently to the CONNJUR integrated environment [26] aiming to support the entire process of molecular structure determination.

3. Requirements analysis and concept development

Unlike the common "all-to-all" format converter approach discussed in Section 2, our solution will be rather based on a uniform approach for low-effort integration of heterogeneous applications and data. Applying this concept, data and interfaces will be created and extended incrementally for each step throughout the workflow without the need to adapt the steps to different data storage types or formats. To systematically develop a concept we have considered all lessons learned from previous work and the requirements from various research communities dealing with multiscale modeling. In the following we will outline these aspects.

First of all, the framework should be generic and domain-independent so that it can be used in diverse multiscale modeling applications with no further modification. Furthermore, the framework should provide a simple meta-modeling environment allowing creation of data models for domain-specific needs. Importantly, the framework should act as a bridge for access to heterogeneous and distributed storage from distributed workflow applications (typically multiscale models) thus providing two interfaces – one for the storage and one for the application side. Crucial capability of the framework should be fully automatic code generation, because only in this case it will provide a low-effort and low-threshold solution for the end-user. In addition, the framework has to inter-operate with grid and cloud middleware deployed on productive e-infrastructures.

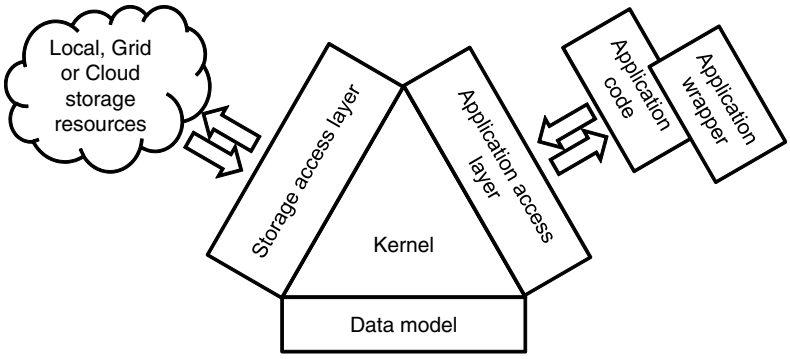

Fig. 1. Conceptual view of the framework

The general logic of the resulting framework is visualized in the form of a triangle as shown in Fig. 1. The central component is the kernel which contains the whole logic for providing a connection between the interfaces of the schema (corresponding to the edges of the triangle in the representation in Fig. 1). The framework is based on a data model that a scientist may freely specify. For example, for simulations of molecular structures, a model can be designed, in which the atoms and their relationships (e.g. chemical bonds) and larger structures of multiple atoms and molecules can be represented. The individual data elements have different properties (e.g. for atoms such as element type, charge, coordinates, etc.). A graphical editor should enable the researcher to define the data model in a simple way independent of the data model complexity. For this a standard so-called meta-model, for example Unified Modeling Language (UML, www.uml.org) can be used providing a domain-independent abstraction, so that the framework can be employed in diverse domains of computational science. From the data model all other components of the framework should be generated automatically in a "push-button" manner, without having to make manual adjustments.

On the left side of the diagram, the storage access layer is intended to enable saving data elements in any available storage resources, e.g. relational databases, simple files or cloud storage services such as Amazon S3. Obviously, the resources may be local or distributed on remote systems and require an abstract implementation of the persistence component to make the access from the application to the data transparent.

The remaining generated framework component is the application access layer through which an application implemented in any arbitrary programming language obtains transparent access to stored data to create, read, update or delete data elements. Moreover, the application access layer should provide interfaces for both local and remote access; for the latter appropriate transmission protocols should be used and security mechanisms for access protection should be integrated.

4. Implementation

In order to demonstrate the benefits of the developed framework we implemented a working prototype. A more detailed discussion focusing on the technical implementation of the framework, including the security aspects, can be found in another paper [27]. Here we limit the discussion therefore only to the corner marks of the implementation.

Because simulation codes in a workflow are written in diverse languages (Fortran, C++, Python etc.) providing interfaces separately for all languages would increase the development and maintenance efforts dramatically. Therefore, our implementation provides language-independent interfaces for storage access, application access and for meta-modeling. For rapid enabling of a broad range of communities and ensuring sustainability the implementation will be distributed under an open source license.

Following the concept introduced in the previous section the implementation of the framework aims at high reuse of existing standards and frameworks and has several components. The first component is a self-contained server application which implements the logic of the kernel and is deployed as a web server. For the implementation of the server application we used platform-independent Java as programming language and further

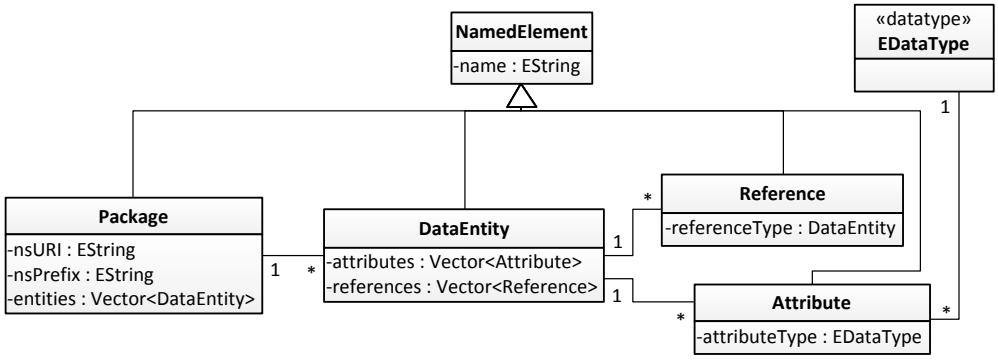

Fig. 2. The custom meta-model employed in the implementation is derived from the Ecore meta-model and contains only elements and properties relevant for data modeling.

technologies and frameworks available for the Java programming environment.

4.1. Modeling

Because different data models can be specified in the framework a so-called meta-model is required, based on which domain-specific models are built. Such meta-models describe the structure and the construction of models an example being the UML meta-model. However, UML is a very complex standard focusing on object-oriented programming. The UML class diagrams, for example, include many aspects which are not needed for data models and even difficult to handle. A lightweight alternative of UML is the Ecore meta-model which is part of the Eclipse Modeling Framework (EMF) [28]. Ecore is limited to object models, similar to UML class diagrams and designed for use in a Java environment simplifying the interaction with models through the automatic generation of Java interfaces to specific models.

Nevertheless, the Ecore meta-model is still too complex for the application because many of the included properties are not used. For instance, the component eOperation representing a function is not relevant for designing pure data models. Therefore, we specified a new simpler meta-model still basing on Ecore [29] but including only components required for data modeling, i.e. EClass, EAttribute, EReference und EDataType (cf. Fig. 2). It consists of four EClass elements: NamedElement, Package, DataEntity, Attribute and Reference. NamedElement is the uppermost class of all other elements and only contains a Name attribute to give each element a name. DataEntities are bundled in a package with a unique Universal Resource Identifier (URI). A DataEntity is an entity that represents a data element such as “atom” and contains attributes and references. Attributes are properties with an Ecore data type (eDataType) and references (Reference) are relationships between entities (cf. Fig. 2). They contain additional properties, such as information about cardinality and nature of relationship. This meta-model forms the basis for modeling and subsequent code generation.

For creation of data models using the meta-model introduced above we constructed a stand-alone editor application basing on the Eclipse Modeling Framework (EMF). The EMF helped to complete this task starting from the definition of the meta-model up to the generation of the program code of a stand-alone graphical editor application. The user interface of the editor is shown in Fig. 3.

4.2. Application access layer

In order to integrate the data model in real-life workflow applications, written in diverse programming languages, it is beneficial considering a service-oriented architecture (SOA), in particular web services, for the implementation of the application access layer. This allows the components of a workflow application to access the data instantly without the need to manage low-level storage of different types and locations because the storage resource might be remote or even distributed over a network. On the other hand, the methods handling the data access in the application are limited to the four persistence storage operations, i.e. create, read, update and delete data elements. The remaining data-specific methods are implemented in the internal logic of the applications and are not part of the application access layer. Therefore, architectural styles for web services such as the REST

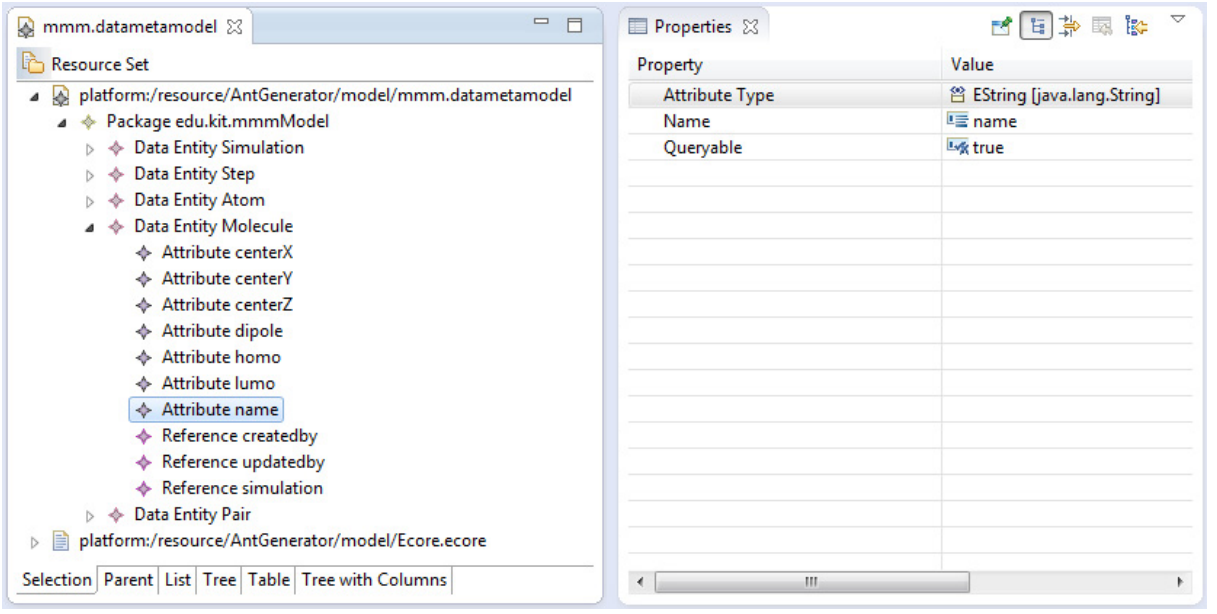

Fig. 3. Graphical model editor application, that has been developed for the Eclipse IDE, enables scientists to construct domain-specific data models.

(Representational State Transfer) [30] are more favorable than the complex SOAP/WSDL standards. A key characteristic of RESTful services is the principle of resources. A resource is a service that is marked by an address or a URI. Using RESTful services the applications interact with resources by means of the HTTP protocol including the basic methods GET, PUT, POST and DELETE. A RESTful service can provide a WADL (Web Application Description Language) document describing the service.

In the Java environment, the Java API for RESTful Web Services (JAX-RS) is a specification with which RESTful services can be created systematically. From several existing implementations of JAX-RS here we employed the reference implementation developed in the Jersey project (https://jersey.java.net/) for the implementation of the application access layer and to create an XML Schema Definition (XSD) file containing the construction plan of the resource representations (see Section 4.4 below).

4.3. Storage access layer

Furthermore the framework needs a technology for persistent storage of the data objects. The primary criterion was the independence of the selected technology of the various storage solutions. This means that any data sources can be used and interchanged in a flexible way. In this respect, the Java Data Objects (JDO) technology seems to provide the optimal solution. The framework DataNucleus selected in this work is the reference implementation for JDO and therefore supports the most recent JDO specifications. Furthermore, DataNucleus is available as open source and provides connections to a large number of different types of data storage.

4.4. Resource representation

To complete the implementation of the framework we need to define a resource representation format used to transport the data between the workflow application and the application layer in the HTTP bodies of GET, POST and PUT requests or responses. We decided to use XML as representation format because XML readers and writers are available for many programming languages. The Java Architecture for XML Binding (JAXB) provides an API that inter-operates with JAX-RS and JDO on the server side and is used here for binding XML data to Java objects and vice versa.

4.5. Automatic code generation, deployment and service infrastructure

For automated generation of the server application code we used the framework Acceleo which supports a template language which allows processing of Ecore models. Using Acceleo, the Java classes required for the remaining components, i.e. storage layer and application access layer, are generated from the data model and then the Java classes are combined with the classes from the DataNucleus (JDO), Jersey (JAX-RS) and JAXB frameworks. The internal program architecture, containing a resource layer, representation layer and persistence layer, as well as the workflow for code generation, are described more precisely elsewhere [27]. Eventually the compiled classes are packed into a WAR file. For setting up a service infrastructure we installed a storage back-end and a Jetty web server to fulfill the service requests by the workflow clients. As a storage back-end we employed a PostgreSQL database, an object-relational database management system supported by JDO.

5. Application integration

To create a new workflow using the framework the only manually written lines of code are for accessing the REST data service over HTTP and to handle the XML resource representations internally. For both tasks APIs are available in many languages. The classes for accessing the REST service can be generated using the provided WADL service description and built into the workflow application steps as client layer enabling the workflow steps to communicate via the HTTP protocol with the REST data service. In the Java programming environment the wadl2java tool can be used. To implement the client layer in shell scripts the cURL program (http://curl.haxx.se/) can be used. The latter approach for a GET request is shown in the following listing:

```
curl -XGET -H 'Content-type: application/xml' \
      http://192.168.2.2/mmm-model/simulation
```

Moreover, the provided about 40 different language and platform bindings of the libcurl API can be used to integrate a client layer for the REST service in diverse other applications. The following listing illustrates the XML representation of a simulation resource as delivered by the REST service in response to a GET request:

```
<?xml version="1.0" encoding="utf-8"?>
<Simulation xmlns="edu.kit.mmmmodel">
  <uri>http://192.168.2.2/mmmmodel/simulation/ff8080813b85b7f8013b890e4fa7006d</uri>
  <id>ff8080813b85b7f8013b890e4fa7006d</id>
  <name>Material Simulation</name>
  <startDate>2012-12-11T08:21:54.320Z</startDate>
</Simulation>
```

Handling of XML is facilitated by tools (of the style readers/writers) for many programming languages (e.g. Java, C++, Python, Perl, Fortran and others) which enable applications written in these languages to directly operate on the XML resource representations internally. For example using the JAXB API, for Java applications there is no additional coding effort because the Java classes containing the data can be automatically generated using the provided XSD schema. Using the REST data service eliminates the need to write code for opening and parsing files and to manage file metadata in the applications. The REST service allows applications to apply selection criteria to resources in GET requests and by this means to perform searches and data extraction. Furthermore, the provided XSD description allows automatic data validation of the against the schema.

6. Case study as proof of principle

In this section we will demonstrate the benefits of the framework employing it in combination with a multiscale model [31] to calculate the charge mobility in an amorphous molecular layer of aluminum tris(8-hydroxyquinoline) (Alq3), a promising material for emitting/electron transport layers in organic light emitting diodes. The present application implements and automates this multiscale model allowing to study the influence of the morphology on device properties from first principles [4, 5]. Because of the predictive power of the model this application can be used routinely in a series of simulations to perform screening of a huge number of possible compounds, morphology and external parameters such as temperature, applied voltage etc.

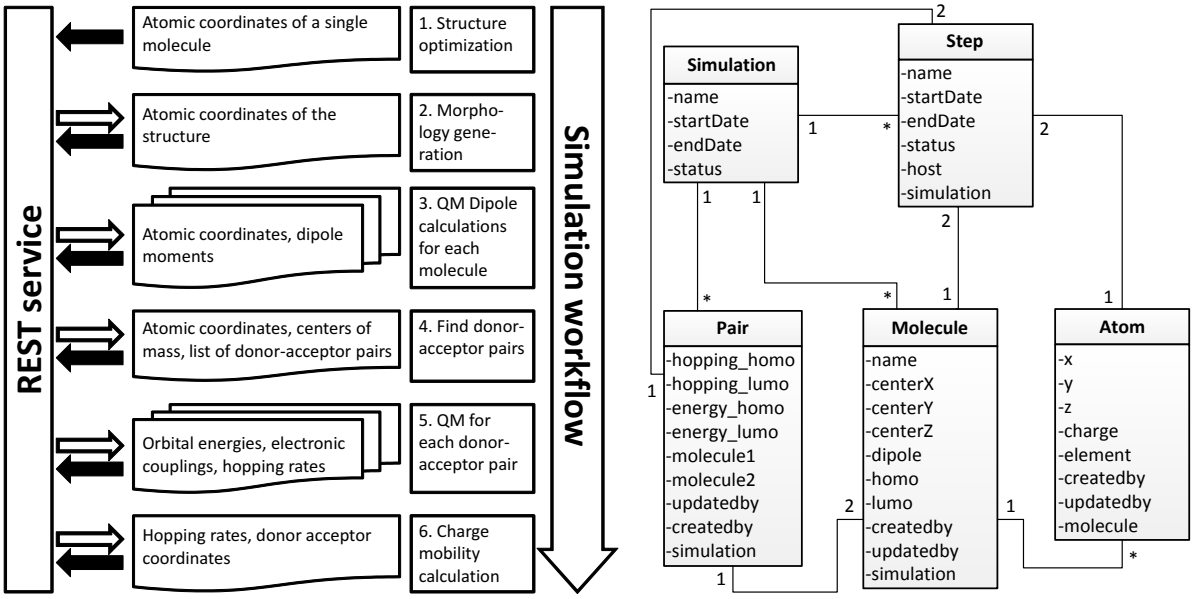

Fig. 4. Workflow of the charge transport simulation (left diagram) and the corresponding data model (right diagram)

The workflow for the charge mobility calculation includes several steps as seen in Fig. 4 in the left diagram. The various data that are passed between the steps were analyzed and mapped onto a data model depicted in the right digram of Fig. 4. This model was then used to generate the REST service which was then used in the simulations using the model. The first workflow step is the structure optimization of a single Alq3 molecule, performed either on a quantum mechanical (QM) level using density functional theory (DFT) or semi-empirical models, yielding the optimized geometry of Alq3. QM approaches are very accurate but costly. Thus, for morphology deposition (step 2) of a molecular layer containing more than a few molecules other methods have to be used such as Molecular Dynamics (MD) or Monte Carlo (MC) methods. In this case study we choose DEPOSIT, a Monte Carlo program with force field parameterization. After step 2 has finished the obtained morphology, i.e. the Cartesian atomic coordinates of all deposited molecules, is sent into the storage using the REST service.

The next step 3 carries out QM calculations of properties, such as dipole moments, for each molecule in the layer with and without the environment of the surrounding molecules. After the single molecules are calculated the Pairfinder in step 4 searches for donor-acceptor pairs of molecules needed for the calculation of the charge-transfer rates (hopping rates). Pairfinder is a Fortran program giving a good example for benefits of the data model framework. There is no standardized file format for the data used by this program, e.g. the centers of masses of the molecules and a cut-off radius. Previously, a special file containing this data has been passed via workflow storage and parsed by Pairfinder [4, 5]. Now, using the framework the data is directly checked out from the storage and the essential input for Pairfinder is the URI of the stored simulation data. In addition, there is no need to write an extra converter from CML or MOL2 formats as has been done previously [4, 5].

After the donor-acceptor pairs have been found (approximately 4000 pairs for the present case) a QM calculation for each pair is carried out (step 5 in Fig. 4, left diagram) yielding the energy difference between the lowest unoccupied molecular orbital (LUMO) of the acceptor and the highest occupied molecular orbital (HOMO) of the donor and the electronic coupling for each pair. This information is extracted “on the fly” from the output files of the QM application (in this case TURBOMOLE) and checked into the simulation storage via the REST service. Previously, the output data from all pair calculations had to be transferred as thousands of separate files to a central collection point (workflow storage) and then combined in a separate workflow step by a combination script [5]. Now, this data is stored using the data model framework and is accessed from the next steps without the need of a central combination script. In the last step 6 of the simulation the charge mobility is calculated as function of applied voltage using the data of the previous steps and the kinetic Monte Carlo method implemented

in the ToFeT program by simulating a time-of-flight experiment for the injected charges. We emphasize that no file format converters were necessary for any step of the workflow because in the framework no files are used.

7. Conclusion

Addressing a major challenge in multiscale modeling, this work provides a data-model driven, platform- and language-independent and service-oriented framework which provides a transparent connection between distributed heterogeneous applications and data storage resources. The framework includes a generic meta-model-based editor for construction of arbitrary data models and can be used in other, completely different application domains. Using the data model as input, the code of the kernel and of the interfaces to storage resources and applications is automatically generated and packed as a Java web application. A novelty in this work is the provision of tools for creation of model-driven generic data services that can be readily employed in diverse application domains to solve real-life problems of multiscale modeling.

Already in the design concept we took into account our previous experience with multiscale modeling and requirements from the community. For the implementation we considered web services and other standards and developed a framework that outperforms other already existing approaches. The framework can be adopted in a scientific environment with high complexity of data and algorithms such as workflow computing environments and particularly useful in various application domains of computational science and multiscale modeling. This recommendation is supported by our first experience with the framework prototype which worked for a complex workflow application in the domain of multiscale materials modeling: simulation of charge transport in organic light emitting diodes. Besides offering several practical and performance benefits the framework will become indispensable for a rapid and cost-efficient development of larger and more complex workflow applications. Potential users of the framework will greatly benefit from its standards conformity by deploying it as middleware in productive e-infrastructures, e.g. in computing and storage grids or clouds. It could also be used in distributed workflow environments for processing and analysis of complex experimental data.

Future work will include the aspects of scalability in the use case of large data, integration of standard security and monitoring components and adding capability to perform data model updates during operation. These improvements can be achieved based on the current implementation version without changing the framework concept because they have been considered already during the design. While easing the technical burden on managing models and data the framework gives the domain expert full freedom in model design and thus it can be combined with ontology-based frameworks such as the Apache Jena framework (jena.apache.org).

Acknowledgements

This work has been partially funded by the 7th Framework Programme of the European Commission within the Research Infrastructures with grant agreement number RI-261594, project MMM@HPC.

References

[1] T. Karakasidis, C. Charitidis, Multiscale modeling in nanomaterials science, Materials Science and Engineering: C 27 (5-8) (2007) 1082–1089. doi:10.1016/j.msec.2006.06.029.

[2] J. A. Elliott, Novel approaches to multiscale modelling in materials science, International Materials Reviews 56 (4) (2011) 207–225. doi:doi:10.1179/1743280410Y.0000000002.

[3] W. E, J. Lu, Multiscale modeling, Scholarpedia 6 (8) (2011) 11527. doi:10.4249/scholarpedia.11527.

[4] I. Kondov, R. Maul, S. Bozic, V. Meded, W. Wenzel, UNICORE-Based Integrated Application Services for Multiscale Materials Modelling, in: M. Romberg, P. Bala, R. Müller-Pfefferkorn, D. Mallmann (Eds.), UNICORE Summit 2011 Proceedings, 7-8 July 2011, Torun, Poland, Vol. 9 of IAS Series, Forschungszentrum Jülich GmbH Zentralbibliothek, Jülich, 2011, pp. 1–10.

[5] S. Bozic, I. Kondov, V. Meded, W. Wenzel, UNICORE-Based Workflows for the Simulation of Organic Light-Emitting Diodes, in: V. Huber, R. Müller-Pfefferkorn, M. R. Romberg (Eds.), UNICORE Summit 2012 Proceedings, May 30-31, 2012, Dresden, Germany, Vol. 15 of IAS Series, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Jülich, 2012, pp. 15–25.

[6] M. Carpené, S. Bozic, I. Kondov, A. Emerson, Experience with UNICORE Services for Multiscale Materials Modelling, in: V. Huber, R. Müller-Pfefferkorn, M. R. Romberg (Eds.), UNICORE Summit 2012 Proceedings, May 30-31, 2012, Dresden, Germany, Vol. 15 of IAS Series, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Jülich, 2012, pp. 1–13.

[7] S. Bozic, I. Kondov, Dataflow Management: A Grand Challenge in Multiscale Materials Modelling, in: P. Cunningham, M. Cunningham (Eds.), eChallenges e-2012 Conference Proceedings, IIMC International Information Management Corporation, 2012, article nr. 38.

[8] R. Ratering, A. Lukichev, M. Riedel, D. Mallmann, A. Vanni, C. Cacciari, S. Lanzarini, K. Benedyczak, M. Borcz, R. Kluszczynski, P. Bala, G. Ohme, GridBeans: Supporting e-Science and Grid Applications, in: Second IEEE International Conference on e-Science and Grid Computing, 2006 (e-Science '06), 2006, pp. 45–52. doi:10.1109/E-SCIENCE.2006.261129.

[9] B. Demuth, B. Schuller, S. Holl, J. Daivandy, A. Giesler, V. Huber, S. Sild, The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows, in: IEEE Sixth International Conference on e-Science (e-Science), 2010, pp. 238–245. doi:10.1109/eScience.2010.42.

[10] S. Sild, U. Maran, M. Romberg, B. Schuller, E. Benfenati, OpenMolGRID: Using Automated Workflows in GRID Computing Environment, in: P. Sloot, A. Hoekstra, T. Priol, A. Reinefeld, M. Bubak (Eds.), Advances in Grid Computing - EGC 2005, Vol. 3470 of Lecture Notes in Computer Science, Springer, 2005, pp. 464–473. doi:10.1007/11508380_48.

[11] S. Sild, U. Maran, A. Lomaka, M. Karelson, Open Computing Grid for Molecular Science and Engineering, J. Chem. Inf. Modeling 46 (2006) 953–959. arXiv:http://pubs.acs.org/doi/pdf/10.1021/ci050354f, doi:10.1021/ci050354f.

[12] W. Dubitzky, D. McCourt, M. Galushka, M. Romberg, B. Schuller, Grid-enabled data warehousing for molecular engineering, Parallel Computing 30 (9-10) (2004) 1019–1035. doi:10.1016/j.parco.2004.07.009.

[13] A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, S. Holl, V. Huber, N. Lamla, D. Mallmann, A. Memon, M. Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, W. Ziegler, UNICORE 6 — Recent and Future Advancements, Annals of Telecommunications 65 (2010) 757–762. doi:10.1007/s12243-010-0195-x.

[14] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, F. Leymann, SIMPL - A Framework for Accessing External Data in Simulation Workflows, in: G. für Informatik (GI) (Ed.), Datenbanksysteme für Business, Technologie und Web (BTW 2011), 14. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 02.-04. März 2011, Kaiserslautern, Germany, Vol. 180 of Lecture Notes in Informatics (LNI), 2011, pp. 534–553.

[15] P. Murray-Rust, H. S. Rzepa, M. Wright, Development of chemical markup language (CML) as a system for handling complex chemical content, New J. Chem. 25 (2001) 618–634. doi:10.1039/B008780G.

[16] P. Murray-Rust, J. Townsend, S. Adams, W. Phadungsukanan, J. Thomas, The semantics of Chemical Markup Language (CML): dictionaries and conventions, Journal of Cheminformatics 3 (1) (2011) 43. doi:10.1186/1758-2946-3-43.

[17] K. Rasch, R. Schöne, V. Ostropytskyy, H. Mix, M. Romberg, The Chemomentum Data Services — A Flexible Solution for Data Handling in UNICORE, in: E. César, M. Alexander, A. Streit, J. L. Träff, C. Cérin, A. Knüpfer, D. Kranzlmüller, S. Jha (Eds.), Euro-Par 2008 Workshops - Parallel Processing, Vol. 5415 of Lecture Notes in Computer Science, Springer, 2009, pp. 84–93. doi:10.1007/978-3-642-00955-6_11.

[18] A. Yang, W. Marquardt, An ontological conceptualization of multiscale models, Computers & Chemical Engineering 33 (4) (2009) 822–837. doi:10.1016/j.compchemeng.2008.11.015.

[19] Y. Zhao, C. Jiang, A. Yang, Towards computer-aided multiscale modelling: An overarching methodology and support of conceptual modelling, Computers & Chemical Engineering 36 (2012) 10 – 21. doi:10.1016/j.compchemeng.2011.06.010.

[20] G. Manduchi, F. Iannone, F. Imbeaux, G. Huysmans, J. B. Lister, B. Guillerminet, P. Strand, L.-G. Eriksson, M. Romanelli, A universal access layer for the Integrated Tokamak Modelling Task Force, Fusion Engineering and Design 83 (2-3) (2008) 462–466. doi:10.1016/j.fusengdes.2007.08.021.

[21] P. Strand, I. Plasencia, B. Guillerminet, F. Imbeaux, M. Haefele, E. Sonnendrucker, R. Coelho, J. Cela, A. Soba, D. Coster, A. Jackson, L.-G. Eriksson, J. Westerholm, F. Iannone, M. Plociennik, M. Owsiak, G. Manduchi, A european infrastructure for fusion simulations, in: 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2010, 2010, pp. 460–467. doi:10.1109/PDP.2010.33.

[22] W. F. Vranken, W. Boucher, T. J. Stevens, R. H. Fogh, A. Pajon, M. Llinas, E. L. Ulrich, J. L. Markley, J. Ionides, E. D. Laue, The CCPN data model for NMR spectroscopy: Development of a software pipeline, Proteins: Structure, Function, and Bioinformatics 59 (4) (2005) 687–696. doi:10.1002/prot.20449.

[23] R. H. Fogh, W. Boucher, J. M. C. Ionides, W. F. Vranken, T. J. Stevens, E. D. Laue, MEMOPS: Data modelling and automatic code generation, J. Integr. Bioinform. 7 (3) (2010) 123–145. doi:10.2390/biecoll-jib-2010-123.

[24] O. Schneider, R. H. Fogh, U. Sternberg, K. Klenin, I. Kondov, Structure Simulation with Calculated NMR Parameters — Integrating COSMOS into the CCPN Framework, in: S. Gesing, T. Glatard, J. Krüger, S. D. Olabarriaga, T. Solomonides, J. C. Silverstein, J. Montagnat, A. Gaignard, D. Krefting (Eds.), HealthGrid Applications and Technologies Meet Science Gateways for Life Sciences, Vol. 175 of Studies in Health Technology and Informatics, IOS Press, 2012, pp. 162–172. doi:10.3233/978-1-61499-054-3-162.

[25] H. Ellis, S. Fox-Erlich, T. Martyn, M. Gryk, Development of an Integrated Framework for Protein Structure Determinations: A Logical Data Model for NMR Data Analysis, in: Third International Conference on Information Technology: New Generations, 2006. ITNG 2006., 2006, pp. 613–618. doi:10.1109/ITNG.2006.52.

[26] R. Nowling, J. Vyas, G. Weatherby, M. Fenwick, H. Ellis, M. Gryk, CONNJUR spectrum translator: an open source application for reformatting NMR spectral data, Journal of Biomolecular NMR 50 (2011) 83–89. doi:10.1007/s10858-011-9497-1.

[27] A. Bender, S. Bozic, I. Kondov, An EMF-based toolkit for the creation of domain-speciffc data services, manuscript submitted (2013).

[28] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks, EMF: Eclipse Modeling Framework, 2nd Edition, Eclipse Series, Addison-Wesley Professional, 2008.

[29] S. Schreier, Modeling RESTful applications, in: Proceedings of the Second International Workshop on RESTful Design, WS-REST '11, ACM, New York, NY, USA, 2011, pp. 15–21. doi:10.1145/1967428.1967434.

[30] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. thesis, University of California, Irvine (2000).

[31] J. J. Kwiatkowski, J. Nelson, H. Li, J. L. Bredas, W. Wenzel, C. Lennartz, Simulating charge transport in tris(8-hydroxyquinoline) aluminium (Alq3), Phys. Chem. Chem. Phys. 10 (2008) 1852–1858. doi:10.1039/B719592C.