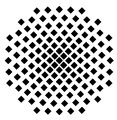

Universität Stuttgart

SimTech Cluster of Excellence

M. Reiter U. Breitenbücher D. Karastoyanova O. Kopp

Quality of Data Driven Simulation Workflows

Stuttgart, October 2012

Institute of Architecture of Application Systems (IAAS)
University of Stuttgart,
Universitaetsstrasse 38
70569 Stuttgart, Germany
{reiter, breitenbuecher, karastoyanova, kopp}@iaas.uni-stuttgart.de
http://www.iaas.uni-stuttgart.de

Abstract Simulations are characterized by long running calculations and complex data handling tasks accompanied by non-trivial data dependencies. The workflow technology helps to automate and steer such simulations. Quality of Data frameworks are used to determine the goodness of simulation data, e.g., they analyze the accuracy of input data with regards to the usability within numerical solvers. In this paper, we present generic approaches using evaluated Quality of Data to steer simulation workflows. This allows for ensuring that the predefined requirements such as a precise final result or a short execution time will be met even after the execution of simulation workflow has been started. We discuss mechanisms for steering a simulation on all relevant levels – workflow, service, algorithms, and define a unifying approach to control such workflows. To realize Quality of Data-driven workflows, we present an architecture realizing the presented approach and a WS-Policy-based language to describe Quality of Data requirements and capabilities.

Keywords Controlling by quality of data; Simulation workflows; Workflow management system; e-science.

Reference Reiter, M.; Breitenbücher, U.; Karastoyanova, D.; Kopp, O. (2012) Quality of Data Driven Simulation Workflows. In: Proceedings of the 8th IEEE International Conference on e-Science, IEEE Computer Society.

Stuttgart Research Centre for Simulation Technology (SRC SimTech)

SimTech – Cluster of Excellence
Pfaffenwaldring 7a, 70569 Stuttgart
publications@simtech.uni-stuttgart.de
www.simtech.uni-stuttgart.de

Quality of Data Driven Simulation Workflows

Michael Reiter, Uwe Breitenbücher, Oliver Kopp, Dimka Karastoyanova

Institute of Architecture of Application Systems (IAAS), University of Stuttgart
{reiter, breitenbuecher, kopp, karastoyanova}@iaas.uni-stuttgart.de

***Abstract*—Simulations are characterized by long running calculations and complex data handling tasks accompanied by non-trivial data dependencies. The workflow technology helps to automate and steer such simulations. Quality of Data frameworks are used to determine the goodness of simulation data, e.g., they analyze the accuracy of input data with regards to the usability within numerical solvers. In this paper, we present generic approaches using evaluated Quality of Data to steer simulation workflows. This allows for ensuring that the predefined requirements such as a precise final result or a short execution time will be met even after the execution of simulation workflow has been started. We discuss mechanisms for steering a simulation on all relevant levels – workflow, service, algorithms, and define a unifying approach to control such workflows. To realize Quality of Data-driven workflows, we present an architecture realizing the presented approach and a WS-Policy-based language to describe Quality of Data requirements and capabilities.**

Key words: controlling by quality of data; simulation workflows; workflow management system; e-science

I. INTRODUCTION

As a type of scientific computations, simulations are characterized by complex calculations and data management tasks. In this paper we describe how simulations can be controlled by Quality of Data (QoD) with the aid of the workflow technology, based on an example Finite Element Method (FEM) [1] simulation. In natural sciences, simulations are usually based on ordinary differential equations (ODEs) or partial differential equations (PDEs) that are solved using numerical methods. A common approach to perform such simulations is transforming differential equations by means of the FEM to a system of matrix equations, which must be solved for every time step.

Workflows are compositions of tasks by means of causal or data dependencies that are carried out on a workflow management system (WfMS) [2]. The workflow technology has been applied in e-science and in particular in simulations to automate execution, to produce traceable results and to reduce errors made by humans [3], [4], [5]. Various scientific workflows exists, some of them support FEM-based simulations [5], [6].

Simulations consume and produce data such as: input data (e.g., spatial information about a simulation object), internally used data (e.g., FEM grid or matrix), intermediate results (e.g., solution vector for an intermediate time step), the final result, output data (e.g., status data) [7]. Some of these types of data have large volumes, complex data structures, and complex dependencies among each other. Hence, the quality of data, together with the quality of algorithms, and computing environments has a strong impact on the final simulation result [7]. In case the QoD is too low, for example the accuracy of the FEM grid is too low for a specific matrix solver, suitable actions such as refining the input data must be carried out to achieve the desired result. In our previous work we have presented methods to determine the QoD within simulation workflows in general [8] as well as in particular within FEM-based simulations [7].

In this paper, we present a QoD-enabled architecture for scientific WfMSs. Therefore, we extend an already existing architecture of a scientific WfMS [5] by (i) a framework that implements the methods to determine the QoD within simulation workflows and (ii) a QoD-driven service bus to select simulation services based on their QoD. This QoD-enabled service bus uses a novel language we introduce in this paper named WS-Quality of Data for defining QoD-Policies. Based on these policies the quality of output data can be ensured (using a quality assurance process) and requirements on the quality of input data for the services selection can be described. Furthermore, we introduce an approach for steering simulation workflows based on QoD – a new trigger for adaptation – across all different layers of a scientific workflows and/or WfMS by matching of assurance and requirement policies.

The rest of this paper is organized as follows: Section II describes the necessary background information. Section III presents QoD mechanisms for workflow navigation, service selection, and service configuration as well as the architecture for a QoD-driven simulation workflow environment. In Section IV we introduce a language for defining QoD-Policies and Section V discusses related work. We conclude the paper and outline future work in Section VI.

II. Background on QoD-driven Simulations

A Simulations

Simulations used in natural sciences that are based on solving ODEs or PDEs, such as FEM-based simulations [1], can be usually divided in three phases, each of them including one or more basic simulation tasks [7]. Figure 1 shows these phases and tasks for an example simulation. Within the preprocessing phase (task 1 to 6) all relevant input data are collected. Based on this input data an initial FEM grid and matrix equation will be generated. In the equation solving phase (task 7 and 8) a matrix equation must be solved and based on intermediate results adapted for all given time steps. In the postprocessing phase (task 9) the simulation results are processed, e.g. visualized. Each simulation task produces or consumes specific types of data. Figure 1 illustrates the important types of data for this example, like the files that will be produced by task 1 (Define Geometry Data) that describe the geometry of the simulation object. These geometry data are input data for tasks 3 through 6 and 8. We observe that a FEM-based simulation is characterized by complex dependencies between the types of data as shown in Figure 1. Reimann et al. [6] describe these dependencies and the appropriate data management operations in detail.

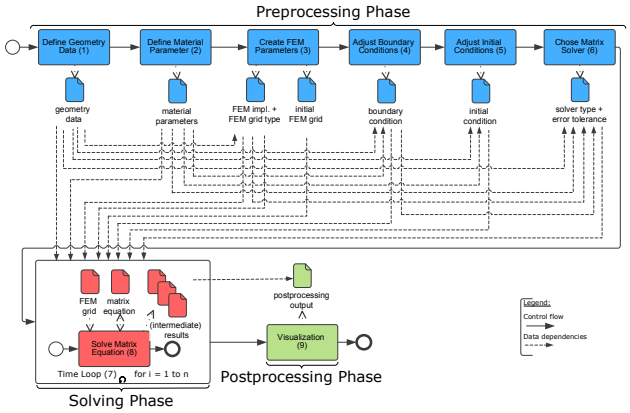

Figure 1: Phases, basic tasks, types of data, and data dependencies of a FEM-based simulation [7].

Simulations consist of three main levels. On the flow level the flow of the tasks can be described, on the algorithm level the algorithm used to implement a task can be defined, and on the parameter level the parameter of the algorithm can be adjusted. Hence, simulations can be influenced on all three levels to meet requirements, e.g., to guarantee a precise final result or a short execution time. On the flow level, the flow of the tasks can be adapted. For instance, if at task 3 the quality of the initial FEM grid is too low to guarantee proper simulation results, task 1 can be executed again to get a smoother geometry of the simulation object to guarantee an initial FEM grid with a better quality. On the algorithm level several algorithms can be chosen to implement a task. For example, at task 8 matrix solvers with a different numerical behavior can be implemented to satisfy the requirements of a specific simulation instance, e.g. on accuracy or execution time if one solver offers a higher accuracy the other a faster execution time [7]. On the parameter level, parameter of the algorithm on the algorithm level can be changed, e.g. at task 3 the granularity of a FEM grid can be adjusted [7].

B Simulation Workflow Management Systems

In recent years, workflow technology has been widely applied to the scientific domain, and several scientific WfMSs have been developed, such as Askalon [9], Kepler [10], Pegasus [11], Taverna [12], Triana [13], and Trident [14]. Görlach et al. [5] present a scientific WfMS using conventional workflow technology [2]. Figure 2 presents the architecture of this WfMS. It makes use of the web service technology as an implementation of the service oriented architecture (SOA) [15] and of BPEL (Business Process Execution Language) [16] as the workflow language. It invokes services to perform simulation tasks via a service bus [17].

Askalon, Kepler, Pegasus, Taverna and Triana support simulation workflows based on data dependencies. Trident and scientific WfMS using conventional workflow technology support simulation workflows based on causal dependencies. More in detail, by using conventional workflow technology a workflow model created within a workflow modeler (see Figure 2) specifies tasks that need to be performed and control flow dependencies between these tasks. This workflow model is a template from which each workflow is instantiated. Hence, a workflow instance is created from a workflow model. This individual workflow instance can be executed by an execution engine (see Figure 2) of a WfMS.

Simulation workflows based on data and causal dependencies are characterized by tasks that perform long running calculations as well as by tasks that perform complex data management operations. Simulation workflows based on conventional workflow technology orchestrate web services as an implementation of tasks. The result is a higher-level, aggregated simulation functionality.

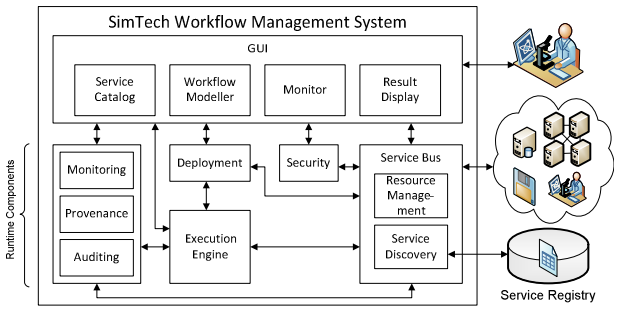

Figure 2: Architecture of a scientific WfMS based on conventional workflow technology [5].

C General Runtime Architecture for Simulation Workflows

The architecture of the workflow runtime environment (see Figure 3 and an alternative representation as part of Figure 2) consists of three conceptual layers. These layers are the ones affected by the QoD aspects.

The *Workflow Management System Layer* is responsible for executing simulation; executing the workflow definitions is also referred to as navigation.

Workflows orchestrate several services which provide sub-functionalities aggregated by the overall simulation process to provide a superior functionality.

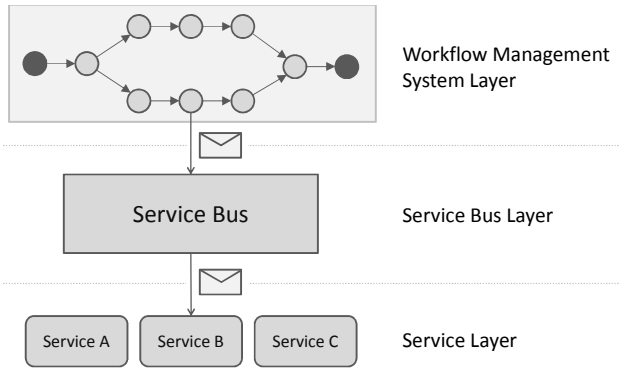

Figure 3: General Runtime Architecture for Simulation Workflows.

The *Service Bus Layer* provides a service bus [17] which is called by the WfMS executing the simulation workflow instances for every service invocation. Thus, simulation workflows do not call services directly but only indirectly by passing each request to the service bus. The bus selects an appropriate service based on functional (e.g. the operation which has to be called) and non-functional requirements (e.g. Quality of Data requirements, response time, etc.), sends the request to the selected service, and returns the response to the initial requestor – i.e. to the requesting workflow instance.

The *Service Layer* describes all services which can be invoked and orchestrated by workflows. Each service provides a standardized interface description [19] of its functionality in terms of operations, parameters, and data types as well as non-functional properties like cost, averaged execution performance indicators, etc. This interface description is used by the service bus to discover appropriate services. Web services describe functionality in terms of WSDL and non-functional properties using WS-Policy [19].

D *Quality of Data*

Different concepts for data quality are established in the business domain, such as methods for measuring quality metrics [20]. In general, QoD consists of six main dimensions to determine the QoD in detail: accuracy, completeness, currency, timeliness, volatility, and consistency [20]. These dimensions can be used in business and in scientific applications [8]. A concept for analyzing QoD impacts on the FEM is presented in [7]. It describes several QoD metrics such as the metric Material Parameter Accuracy or Vector Condition. The first metric is related to task 2 in Figure 1 to verify material parameters of the PDE. The second metric is related to task 8 in Figure 1 to validate the numerical condition of a vector within a matrix equation. For each dimension and metric the QoD can be expressed as a value in the interval [0,1] with 0.0 meaning a bad QoD and 1.0 – a good QoD.

QoD measurement is not a monolithic process so that one could determine in a single step whether data is good or bad [8]. A distinction must be made between data characteristics, e.g., the accuracy of two decimal places of values within a vector, and data goodness, e.g., whether the accuracy of two decimal places is good enough for a matrix solver to compute a result with a sufficient precision required by the overall simulation. In our previous work we have presented a two-step process to determine the QoD [8] (see also Figure 4). This process is has three important principles:

Decoupling data characteristics and the data goodness: Data characteristics with respect to QoD describe well defined characteristic properties of data based on one or more data quality dimensions without a simulation context. In the process to determine the QoD (Figure 4) data characteristics are represented by metrics measured by the measurement task. Data goodness is the result of the interpretation of measured characteristics in a special simulation context. This allows the evaluation of certain characteristic properties depending on the context that can results in different QoD values. For example, an accuracy of two decimal places can have a high QoD for the processing by an algorithm that implements matrix solver 1, but a low QoD for the processing by another algorithm that implements matrix solver 2, although there was only one measured characteristic (e.g., accuracy) for the input data. In this case, the evaluation contexts are associated with an algorithm on solver 1 or an algorithm on solver 2. In principle, there are strong dependencies between measured characteristics of data, evaluation context and the final QoD [8].

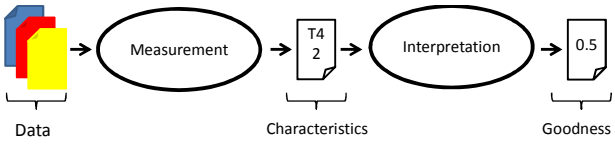

Figure 4: Two-step process to determine QoD [8].

Measurement and interpretation processes: Measurement and interpretation processes are typically implemented via domain specific algorithms. [7] show examples such as an algorithm that measure the vector condition within a matrix solver. A metric measurement process will return metrics as input for possible interpretation processes. For example, a metric measuring the vector condition have the value 0.5. An interpretation gets this value as input and evaluates whether the vector can be used for a certain matrix solver or not by checking if the threshold of 0.4 is exceeded for solver 1 or 0.8 for solver 2.

Subjective and objective determination: In general, measurements and interpretations can be performed automatically by computers in an objective way or manually by scientists in a subjective way. Hence, it must be distinguished between the determination done by software and done by human beings.

We have implemented a corresponding QoD Framework that performs the two step process based on [8]. It allows subjective and objective QoD determination. In Section III and IV implementation details can be found.

III. Levels of Control

In simulations the quality of all types of data results in requirements to steer the execution of the simulations on the flow, algorithm, and parameter level [7]. The same is true for QoD-driven simulation workflows on the corresponding workflow navigation, service selection, and service configuration level described in Section II.

A QoD Key Aspects in Simulation Workflows

Two distinct key aspects, workflow path and quality influence, exist in QoD-driven workflow execution, called in the following *QoD aspects*. On the one hand, QoD can influence the execution of a simulation workflow. On the other hand, the execution of a simulation workflow can influence the QoD.

First, QoD can influence the execution paths of workflows and the choice of orchestrated services. A scenario is the suitability of certain algorithms to process input data having a bad quality in terms of a certain metric. An example is solving a matrix equation. There are solvers which require a high degree of data accuracy to work at all and other solvers which require less quality because their results are not accurate anyway and thus the quality of the input data is not that important [7]. Therefore, depending on QoD values, workflow executions may be different in terms of chosen control flow paths and services.

The second aspect is that the workflow execution may influence the quality of certain data. The first scenario above can be modified as follows: To solve a matrix equation, there are two suitable services fulfilling the functional requirements of the task, i.e., solving a matrix equation of a certain data type according to a certain defined procedure. Both services implement different algorithms, one achieves a higher degree of accuracy than the other but therefore needs more time for execution. Thus, the services differ in the quality of their output data in terms of accuracy and the choice of service for this specific task influences the quality of the data returned to the simulation workflow. As data is processed, typically by multiple activities of a simulation, the quality may be very important to avoid error propagation to further steps [7].

B QoD-driven Simulation Workflow Environment Architecture

We extend the architecture shown in Figure 2 by (i) a Quality of Data framework presented in our previous work [8] and by (ii) a Quality of Data aware service bus (see Subsection D) that supports the WS-Policy standard [18], a specification to express non-functional properties of services and corresponding matchmaking functions. Weerawarana et al. [19] give a conceptual overview of the WS-Policy concept and its usage with the conventional workflow technology. Furthermore, we introduce a WS-Policy-based language called WS-Quality of Data for defining QoD-Assurance-Policies and QoD-Requirement-Policies. Doing so, a simulation workflow can be steered on three levels based on a novel trigger, namely the QoD. Figure 3 present these levels. The flow level corresponds to workflow navigation on the Workflow Management System Layer, the algorithm level with service selection on the Service Bus Layer, and the parameter level with service configuration on the Service Layer.

To enable the QoD aspects, we present three *QoD Mechanisms* which are applied to the components of the runtime architecture: *QoD-based Workflow Navigation*, *QoD-based Service Selection*, and *QoD-based Service Configuration*. The application of these mechanisms requires a refinement of the runtime architecture, from Figure 3. First, the service bus is replaced by a *Quality of Data-driven Service Bus*. Second, the services have to be exposed as *Quality of Data-driven Services* by means of implementing standardized interfaces. These interfaces provide QoD functionalities and non-functional QoD annotations in the form of policies – both used by the QoD-driven Service Bus. Third, a new *QoD Framework* component [8] is added providing a service which offers operations used to calculate quality of passed data in terms of specified measurement and interpretation (see Section II). The resulting architecture is denoted a *Quality of Data-driven Simulation Workflow Environment Architecture* and is shown in Figure 5.

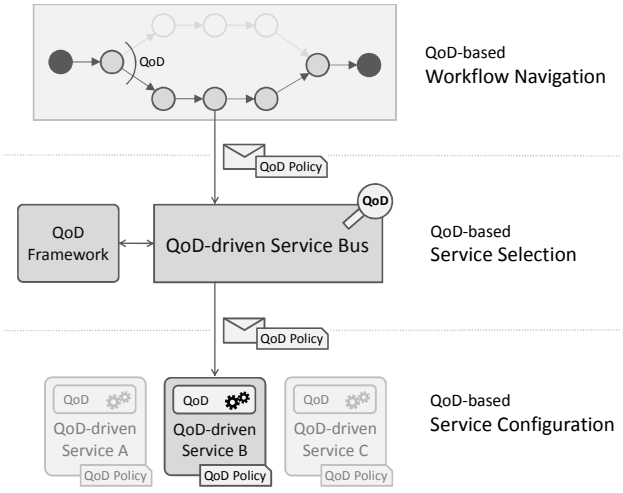

Figure 5: Quality of Data-driven Simulation Workflow Environment Architecture.

C QoD-based Workflow Navigation

Workflows are graphs defining the execution order for a set of activities, as well as data dependencies. Activities implicitly influence the quality of their output data whenever they create or process data. In many cases, the quality of output data depends on the quality of the input data the activities process, which is also valid for sequences of activities in a workflow path. Workflows allow defining conditional paths which are executed only if certain conditions are met (e.g. a defined threshold is reached by a certain variable). This allows also controlling the quality of data by means of explicit control mechanisms directly built into the workflow itself, which is one of the contributions in this work. This way, scientists are able to control QoD by control variables that are used to evaluate which paths have to be taken, i.e., they are used to control the navigation through the workflow. One practical example related to QoD is the simulation execution time. In many cases, scientists want the simulation to be executed only for testing purposes and

therefore fast execution time is preferred and a high quality of data is not absolutely needed. Since the execution time of computation algorithms, such as matrix solvers, mainly depends on the required quality of their output data, these tests are executed with only low requirements for QoD, which means that the algorithms can be executed much faster in order to decrease the overall execution time. In practice, a Boolean variable (a.k.a. transition conditions) may represent the predefined modes, i.e. if the workflow shall take the path(s) which compose algorithms achieving high QoD or if the path(s) achieving low QoD are sufficient enough.

QoD values can be used within the transition conditions in order to control the navigation through the workflow. This is, QoD values determine if a path has to be executed or not. This may be important if certain activities need a minimum QoD to produce meaningful results, in which case lower QoD would be sufficient. In this case the workflow engine needs the values of QoD for each workflow instance. For this the Quality of Data-driven Simulation Workflow Environment Architecture defines a *QoD Framework* component, which provides the so-called *QoD Services* to calculate the quality. These services can be invoked by workflows via the service bus to evaluate data in terms of certain QoD measurements and interpretations (see Section II). The calculated QoD values can then be used as *QoD-based Control-Variables* for conditional navigation based on QoD.

D QoD-based Service Selection

There are multiple algorithms providing identical functionality but differing in the applied mathematical to achieve this functionality, like numerical tasks. Depending on provided QoD of input data and required QoD of results, different algorithms are suitable for certain tasks. On the one hand, there are cases where a bad QoD of input data restricts the use of some algorithms because they are not able to deal with this data at all. On the other hand, requirements on the quality of output data, for example significant decimal places of result values may be required, which reduces the number of suitable algorithms because some of them cannot guarantee the required accuracy. Therefore, there is a need for selecting appropriate algorithms which implement tasks in a simulation workflow under consideration of QoD requirements on both input data and output data.

In the presented QoD-driven Simulation Workflow architecture, all functionalities used by simulation workflows are provided as services to foster integration. The algorithms are implemented as services exposing stable interfaces that describe their functionality. All services are invoked by the workflow via the service bus (see Section II). As the service bus is the central component responsible for selecting and executing the actual service invocations, the bus has full control on which services are called by the workflow and hence we propose to delegate the service bus also the responsibility of taking the QoD into account.

Our architecture tackles the first QoD aspect by introducing a central *QoD-driven Service Bus* which supports QoD-based Service Selection: the selection of services depends not only on functional requirements but also takes non-functional QoD-requirements into account. The bus selects services which are appropriate for the quality of the passed/provided input data and have the capabilities to provide the quality of the output data of the service as required by the requestor. This means that the workflow execution, in particular the selection of services implementing activities, is controlled by QoD. To enable the comparison of services in terms of QoD capabilities, we use the concept of policies. Each service request expresses the requested quality of output data and provided quality of input data in the form of a *QoD-Policy* which is contained in the header of the request. Each service itself describes its QoD capabilities, i.e. which quality of output data it can guarantee, and its QoD requirements, i.e., which quality of input data it needs, for the offered operations within such a QoD-Policy. The service bus in the proposed architecture is responsible for selecting a service whose policy matches with the policy contained in the request. The actual quality of input data is either already provided by the policy of the request, where it was put into by the requestor, i.e., the workflow. Otherwise, e.g., if there are only services requiring a certain minimum quality of input data to be able to process the request at all, it is explicitly calculated by the service bus by invoking the QoD-Framework. In this case, the service bus modifies the QoD-Policy contained in the request by adding the calculated QoD information. We describe the language and mechanisms related to these policies more detailed in Section IV.

E QoD-based Service Configuration

Services providing mathematical functionalities are in most cases customizable and can be parameterized for processing certain requests [1]. For example, a service to create a FEM grid can be adjusted by granularity parameter, which leads to different quality of the FEM grid as output data in terms of accuracy. To enable this, the presented architecture introduces services supporting *QoD-based Service Configuration* via standardized meta-information contained in the header fields of the request used to configure the execution of an operation in terms of QoD. To enable specifying parameters supported by services and how the parameters relate to required quality of input data and guaranteed quality of output data, the QoD-Policy concept introduced in the previous section is used. Each service has QoD-Policies attached to its operations. The policies may define multiple QoD requirements on input QoD and capabilities for output QoD. After the service bus has found a matching policy, the service bus calculates the so-called effective policy [17] containing the agreed upon QoD statements. For example, if a service operation offers two parameters related to different QoD guarantees for its output data, the service bus selects the one which meets the QoD requirements of the requestor and writes this combination in the effective policy. The resulting effective policy is then used as meta-information contained in the request header to configure the service, i.e., to tell the service which parameter it shall apply for processing the current request. After invoking the service, the QoD-driven

Service Bus may validate the results in order to ensure that the promised QoD values are provided.

IV. WS-Quality of Data

In this section we present a WS-Policy-based [18] language named *WS-Quality of Data (WS-QoD)* for defining QoD-Policies used to discover and configure services based on QoD requirements and assurances. The language enables the realization of the two presented mechanisms: (i) QoD-based Service Selection and (ii) QoD-based Service Configuration as presented in the previous sections. Our QoD Framework uses this language.

A Mechanisms

WS-Quality of Data introduces two different Quality of Data statements: *Quality of Data Requirement (QoDR)* and *Quality of Data Assurance (QoDA)*: A QoDR is a formal definition of the required QoD, a QoDA is a formal definition of assured QoD. In interactions between simulation workflows and services both partners can use both statements to express capabilities and requirements: A requestor may express Quality of Data requirements for the output data of the service and can make assurances about the quality of the provided input data. On the other hand, a service can have requirements for the quality of its input data and give assurances about the quality of its output data. Thus, the general mechanism is that each QoD requirement has to be satisfied by a suitable QoD assurance. Both statement types can be combined by a partner to express conditioned dependencies between input and output Quality of Data, e.g. a service may guarantee that an assurance for output QoD only holds if a certain quality of input data is provided. The service discovery has to be capable to find matching QoD statements of requestor and service: For each requirement statement of one partner there has to be a matching assurance statement of the other partner service which meets the requirement.

B Syntax

To express Quality of Data Requirements and Assurances we extend WS-Policy and introduce a new AssertionType called *QualityOfDataAssertion*. Listing 1 shows the meta-model as an XML Schema representation. Each assertion defines its statement type (i.e. requirement or assurance) via a type attribute, the id of the applied QoD measurement or interpretation method (QoDId-Element, e.g. "uni-stuttgart.simtech.qod.metrics.accuracy", Listing 2), the actual QoD value (QoD-Element), and references to the corresponding data which are either contained in input or output variables of the corresponding operation (DataRef-Element). The QoD element allows defining either a certain value, or a minimum value, or a maximum value, or a range (combining minimum value and maximum value) which is required or assured. The DataRef-Element may be extended to support special transport formats and mechanisms between requestor and service such as SOAP over HTTP, SOAP over JMS, etc. As SOAP is an XML-message format, XPath expressions to identify the corresponding data within the message may be used as additional expression type attribute, for example. Assurance Assertions may optionally define a condition containing Requirement Assertions which have to be satisfied to make the main assurance assertion valid. To enable using WS-Policy matchmaking, QoD Assertions of type assurance have to be declared as optional because they are only used to meet requirements. Otherwise, the intersection would have to find a matching requirement for every assurance to be compatible but this is not needed as assurances are only offerings to fulfill requirements.

```
<QoDAssertion type="Requirement | Assurance">
   <QoDId> … </QoDId>
   <QoD>!
      (
         <Value xsi:type="xs:… ">
            …
         </Value>
      |
         <MinimumValue xsi:type="xs:… ">?
            …
         </MinimumValue>
         <MaximumValue xsi:type="xs:… ">?
            …
         </MaximumValue>
      )
   </QoD>
   <DataRef type="Input | Output" name="…" />?
   <Condition>?
      <QoDAssertion type="Requirement" />+
   </Condition>
</QoDAssertion>
```

Listing 1: QualityOfDataAssertion using XML syntax.

```
<QoDAssertion type="Assurance" optional="true">
   <QoDId>
      uni-stuttgart.simtech.qod.metrics.accuracy
   </QoDId>
   <QoD>
      <MinimumValue xsi:type="xs:double">
         0.5
      </MinimumValue>
   </QoD>
   <DataRef type="Output" name="result" />
   <Condition>
      <QoDAssertion type="Requirement">
         <QoDId>
            uni-stuttgart.simtech.qod.metrics.
            completeness
         </QoDId>
         <QoD>
            <MinimumValue xsi:type="xs:double">
               0.9
            </MinimumValue>
         </QoD>
         <DataRef type="Input" name="equation"/>
      </QoDAssertion>
   </Condition>
</QoDAssertion>
```

Listing 2: Example of a QualityOfDataAssertion.

Listing 2 shows an example QoD Assertion defining a conditioned assurance: The assertion assures an accuracy of its output data with value 0.5 only if the input data for the corresponding operation has a completeness of 0.9.

C *Compatibility of QoD Assertions*

In contrast to the standard WS-Policy compatibility check of assertions which only compares QNames, QoD Assertions need to be compared in more detail. Two QoD Assertions are compatible if the following additional rules hold. One assertion has to be of type requirement, the other of type assurance having the same QoDId and compatible DataRef. As the DataRef-Element depends heavily on the used message format (e.g. SOAP), this compatibility check is based on a plugin mechanism to be extensible. The actual QoD values have to be identical if they are defined in a Value-element. If the QoD Requirement Assertion defines a MinimumValue, the QoD Assurance Assertion has to define a MinimumValue or Value which is equal or higher. MaximumValue is equivalent to this. If the assertion is conditioned, the nested QoD Requirement Assertions have to be satisfied to make the assurance assertion valid.

D *Matchmaking and QoD-based Post Processing*

The service discovery process uses domain specific WS-Policy intersection for matchmaking, i.e. to find matching combinations of compatible and valid QoD Assertions. Figure 6 shows this step.

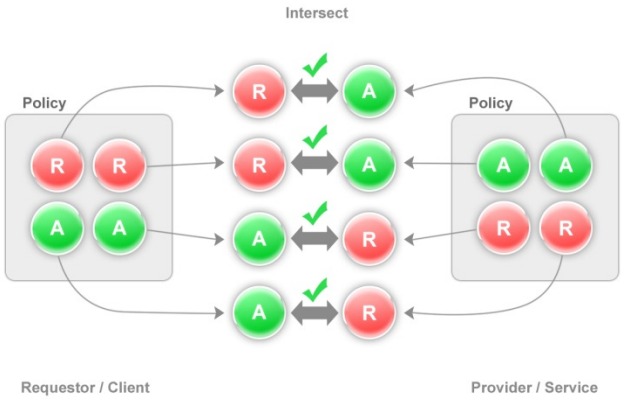

Figure 6: Compatible QoD Assertions in different Policies.

For that, first the standard WS-Policy intersection algorithm is used which identifies QoD Assertions contained in alternatives only by their QNames and merges them into alternatives for creating a precursor of the actual effective policy. As the QNames only represent the AssertionType and not the QoD-domain specific information, there is a need for a QoD-based post processing phase which inspects the resulting policy in order to analyze if the assertions contained in alternatives are compatible: For each QoD Assertion defining a QoD requirement there has to be a QoD Assertion providing the needed assurances in the same alternative, otherwise the QoD requirements are not met and the alternative is not valid for processing the request. Therefore, the domain-specific post processing of the intersection keeps only those QoD Assertions which define requirements and those defining assurances which meet the requirements. All other QoD Assertions providing assurances are removed. If an alternative contains at least one QoD Assertion defining a requirement which is not satisfied by another QoD Assertion in the same alternative, the alternative gets removed from the policy. The result of these two steps is the effective policy which is used for interaction. If this effective policy contains no alternatives, the interaction cannot take place.

If there are multiple alternatives in the effective policy containing compatible QoD Assertions, i.e. valid combinations of requirements and assurances, the QoD-driven Service Bus selects exactly one alternative (which one depends on the bus configuration) and removes all the others. The resulting effective QoD-Policy is put into the request header sent to the service by the service bus (the specification of headers is out of scope of this work).

Listing 3 shows two compatible QualityOfData-Assertions which are contained in an example alternative. The first assertion defines a minimum QoD value requirement which is met by the assurance provided by the second assertion.

```
<QoDAssertion type="Requirement">
   <QoDId>
      uni-stuttgart.simtech.qod.metrics.accuracy
   </QoDId>
   <QoD>
      <MinimumValue xsi:type="xs:double">
         0.4
      </MinimumValue>
   </QoD>
   <DataRef type="Output" name="result" />
</QoDAssertion>

<QoDAssertion type="Assurance">
   <QoDId>
      uni-stuttgart.simtech.qod.metrics.accuracy
   </QoDId>
   <QoD>
      <MinimumValue xsi:type="xs:double">
            0.5
      </MinimumValue>
   </QoD>
   <DataRef type="Output" name="result" />
</QoDAssertion>
```

Listing 3: Two compatible QualityOfDataAssertions.

V. Related Work Discussion

The first attempt to analyze the impact of QoD on the preprocessing, equation solving, and postprocessing phase within FEM-based simulations (see Figure 1) is described in our previous work in [7]. Other research papers depend on a specific application and operation ranges. Thus, these approaches cannot be used in a general sense within FEM-based simulations [7]. Gray et al. [21] call for the control of QoD in simulations without making any concrete implementation suggestion. In our previous work, we introduce a framework to determine QoD in a generic manner [8]. We use this framework as a foundation for our approach presented here to steer simulation workflows by QoD on the workflow navigation, service selection, and service configuration level as described in Section II. Na'im et al. [22] explain an implementation based on Kepler to define data quality threshold values as well as to monitor and visualize QoD findings during runtime. Nevertheless, Na'im et al. do not point out details of how to measure and analyze QoD metrics. Based on Taverna,

Missier et al. [23] describe the Qurator workbench that can use given QoD findings to steer data flow oriented scientific workflows by QoD related threshold values. Qurator does not measure QoD metrics. To the best of our knowledge, we introduce the first approach to steer simulation workflows (based on the conventional workflow technology) using QoD on workflow navigation, service selection, and service configuration level.

VI. Conclusion and Future Work

This paper identifies three approaches for steering Quality of Data (QoD) driven simulation workflows: on the workflow level (workflow navigation), on service level (service selection), and related to algorithms used (configuration by parameters). We unify these approaches to a common approach to steer simulation workflows on all levels by QoD requirements and QoD assurances on input data as well as on output data. For this, we have extended the architecture of conventional WfMSs to a so called Quality of Data driven Simulation Workflow Environment Architecture and defined a WS-Policy-based language for Quality of Data.

QoD determination can be performed by software or manually by humans. Therefore, our future work will focus on tools and best practices to support scientists to find quality characteristics within data and interpret these characteristics in a simulation specific context. Furthermore, we will concentrate on the research of QoD-driven multi-scale simulation workflows, e.g., to analyze the dependencies between QoD on the higher scale and the QoD on the lower scale.

Acknowledgement

The work presented in this paper has partly been funded by the DFG Cluster of Excellence Simulation Technology (http://www.simtech.uni-stuttgart.de) (EXC310) and by the BMWi project CloudCycle (01MD11023).

References

[1] Zienkiewicz, O.; Taylor, R.; Zhu, J.: The Finite Element Method – Its Basis & Fundamentals. 6th Edition. Elsevier Ltd. Butterworth-Heinemann (2005).

[2] Leymann, F.; Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall, Englewood Cliffs, NJ (1999).

[3] Taylor, I.; Deelman, E.; Gannon, E.: Workflows for e-Science – Scientific Workflows for Grids. Springer, London, UK (2007).

[4] Hey, T., Tansley, S., Tolle, K.: The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft, Redmond, WA (2009).

[5] Görlach, K.; Sonntag, M.; Karastoyanova, D.; Leymann, F.; Reiter, M.: Conventional Workflow Technology for Scientific Simulation. In: Guide to e-Science, Springer (2011).

[6] Reimann, P.; Reiter, M.; Schwarz, H.; Karastoyanova, D.; Leymann, F.: SIMPL – A Framework for Accessing External Data in Simulation Workflows, In Proc. 14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (2011).

[7] Reiter, M.; Dustdar, S.; Karastoyanova, D.; Krause, R.; Leymann, F.; Pahr, D.; Truong, H.: On Analyzing Quality of Data Influences on Performance of Finite Elements driven Computational Simulations. In: European Conference on Parallel Processing (Euro-Par) 2012 – 18th International Conference (2012) (in press).

[8] Reiter, M.; Breitenbuecher, U.; Dustdar, S.; Karastoyanova, D.; Leymann, F.; Truong, H.: A novel framework for monitoring and analyzing quality of data in simulation workflows. In: 7th IEEE e-Science International Conference (2011).

[9] Fahringer, T.; Jugravu, A; Pllana, S.; Prodan, R.; Seragiotto, C.; Truong, H.: ASKALON: a tool set for cluster and Grid computing: Research Articles, Concurrency and Computation: Practice & Experience (2005).

[10] Altintas, I.; Berkley, C.; Jaeger, E.; Jones, M.; Ludascher, B.; Mock, S.: Kepler: An Extensible System for Design and Execution of Scientific Workflows. SSDBM (2004).

[11] Deelman, E.; Blythe, J.; Gil, Y.; Kesselman, C.; Mehta, G.; Patil, S.; Su, M.-H.; Vahi, K.; Livny, M.: Pegasus: Mapping Scientific Workflows onto the Grid. Lecture Notes in Computer Science, Volume 3165/2004, Second European AcrossGrids Conference, Springer, pp. 11-20 (2004).

[12] Oinn, T.; Greenwood, M.; Addis, M.; Nedim Alpdemir, M.; Ferris, J.; Glover, K.; Goble, C.; Goderis, A.; Hull, D.; Marvin, D.; Li, P.; Lord, P.; Pocock, M.R.; Senger, M.; Stevens, R.; Wipat, A.; Wroe, C.: Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. Concurrency and Computation: Practice and Experience, 18(10):1067-110 (2006).

[13] Churches, D.; Gombas, G.; Harrison, A.; Maassen, J.; Robinson, C.; Shields, M.; Taylor, I.; Wang, I.: Programming Scientific and Distributed Workflow with Triana Services. Concurrency and Computation: Practice and Experience. Special Issue on Scientific Workflows (2005).

[14] Barga, R.; Jackson, J.; Araujo, N.; Guo, D.; Gautam, N.; Simmhan, Y.: The Trident Scientific Workflow Workbench. In: Proc. of the 4th International Conference on e-Science, Indianapolis, Indiana (2008).

[15] Papazoglou, M..; Georgakopoulos, D.: 2003. Introduction: Service-oriented computing. Commun. ACM 46, 10 (2003).

[16] Alves, A.; Arkin, A.; Askary, S.; Barreto, C.; Bloch, B.; Curbera, F.; Ford, M.; Goland, Y.; Guízar, A.; Kartha, N.; Liu, C. K.; Khalaf, R.; König, D.; Marin, M.; Mehta, V.; Thatte, S.; van der Rijn, D.; Yendluri, P.; Yiu, A.: Web Services Business Process Execution Language Version 2.0, (2008).

[17] Leymann, F.: The (Service) Bus: Services Penetrate Everyday Life. Proceedings of the 3rd International Conference on Service Oriented Computing (ICSOC), Springer Verlag, (2005).

[18] Vedamuthu, A.; Orchard, D; Hirsch, F.; Hondo, M.; Yendluri, P.; Boubez, T.; Yalçinalp, Ü.: Web Services Policy 1.5 - Framework, (2007).

[19] Weerawarana, S.; Curbera, F.; Leymann, F.; Storey, T.; Ferguson, D.F.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More, Prentice Hall PTR, (2005).

[20] Batini, C.; Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques, ser. Data-Centric Systems and Applications. Springer (2006).

[21] Gray, J.; Liu, D. T.; Nieto-Santisteban, M.; Szalay, A.; DeWitt, D. J.; Heber, G.: Scientific data management in the coming decade, SIGMOD Rec., vol. 34, no. 4, pp. 34–41 (2005).

[22] Na'im, A.; Crawl, D.; Indrawan, M.; Altintas, I.; Sun, S.: Monitoring data quality in Kepler. In: Proc. of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10). ACM, New York, NY, USA (2010).

[23] Missier, P.; Embury, S.; Greenwood, M.; Preece, A.; Jin, B.: Managing information quality in e-science: the qurator workbench. In: Proc. of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD '07). ACM, 1150-1152 (2007).