**Institute of Architecture of Application Systems**

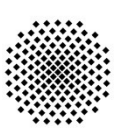
# Migrating e-Science Applications to the Cloud: Methodology and Evaluation

Steve Strauch, Vasilios Andrikopoulos, Dimka Karastoyanova, and Karolina Vukojevic
Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{firstname.lastname}@iaas.uni-stuttgart.de

BIBTEX

```
@INBOOK{Strauch2015,
author = {Strauch, Steve and Andrikopoulos, Vasilios and Karastoyanova,
          Dimka and Karolina Vukojevic},
title = {Migrating e-Science Applications to the Cloud: Methodology and
          Evaluation},
chapter = {5},
pages = {89--114},
series = {Cloud Computing with e-science Applications},
editor = {Oliver Terzo and Lorenzo Mossucca},
publisher = {CRC Press/Taylor \& Francis},
year = {2015}
}
```





**University of Stuttgart**
Germany

*Author Name*

# *Book title goes here*

# *List of Figures*

# *List of Tables*

# *Contents*

# Part I

# This is a Part

# 1

# *Migrating eScience Applications to the Cloud: Methodology and Evaluation*

**Steve Strauch**

*Institute of Architecture of Application Systems (IAAS), University of Stuttgart, Germany, steve.strauch@iaas.uni-stuttgart.de*

**Vasilios Andrikopoulos**

*Institute of Architecture of Application Systems (IAAS), University of Stuttgart, Germany, vasilios.andrikopoulos@iaas.uni-stuttgart.de*

**Dimka Karastoyanova**

*Institute of Architecture of Application Systems (IAAS), University of Stuttgart, Germany, dimka.karastoyanova@iaas.uni-stuttgart.de*

**Karolina Vukojevic-Haupt**

*Institute of Architecture of Application Systems (IAAS), University of Stuttgart, Germany, karolina.vukojevic@iaas.uni-stuttgart.de*

## CONTENTS

## 1.1 Introduction

eScience is an active field of research striving to enable faster scientific discovery and ground-breaking research in different scientific domains by means of information technology. It is considered a new paradigm for science and is referred to as the *fourth paradigm* (Hey et al., 2009) or *data-intensive science* and unifies theory, experiments, and simulation for data exploration for the purpose of scientific discovery. Existing literature shows that there is a myriad of available software systems supporting only some of the experiment life cycle phases and applicable only for specific scientific domains, like Kepler, Triana, Taverna, Pegasus etc. (Taylor et al., 2006).

Due to its interdisciplinary nature, eScience exhibits a high degree of complexity, mainly due to the technical challenges and interoperability deficiencies of the existing software, the large amounts of data produced and consumed by the computational tools and systems, and the computational intensity and distributed characteristics of the IT environment observed in scientific computing. One major issue in current research is the integration of existing software and tools, across domains and organizational structures, for enabling the collaborative modeling of more complex scientific experiments and their execution. The most prominent approach for integrating software systems for the purpose of performing scientific experiments is the workflow technology. Workflows are defined in terms of control flow among tasks comprising an experiment and the data exchanged among them, i.e. data flow. Moreover, the tasks in a workflow stand for a concrete unit of work that can be implemented by a computational, configuration or visualisation tool, or by human users.

The available scientific workflow systems can be classified in two groups based on the fundamental features of the workflows they realize. There are data-driven scientific workflow systems, like Kepler, Triana, Taverna, and Pegasus (Taylor et al., 2006), which stem from the research in scientific computing. In such workflows the focus is on modeling experiments in terms of how scientific data is processed (i.e. the tasks in a workflow are data processing tasks), distributed and placed on computing nodes in terms of computing jobs. There are also control flow-based scientific workflow systems, like SimTech SWfMS[1]and Trident[2], which support workflows with emphasis on the control flow among computational tasks, while the data consumed and produced by the software systems follow the control flow. In these workflows the computational task are implemented by individual software systems, which in turn may distribute the computation over multiple computing nodes, however this is kept transparent for the workflow system. The enacting environment, also called workflow management system or workflow engine, is mainly dealing

[1]SimTech Scientific Workflow Management System: `http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/projects.php`

[2]Trident: `http://research.microsoft.com/en-us/collaboration/tools/trident.aspx`

with orchestrating the software systems, as well as human users. Such workflows have been developed as extensions to the available workflow technology from business applications.

These two different types of scientific workflow systems exhibit very different quality of service characteristics like scalability, robustness, interoperability, reusability, and flexibility (De Roure et al., 2009), (Görlach et al., 2011), (Sonntag and Karastoyanova, 2010). The systems based on the conventional workflow technology from the business domain exhibit better quality of service characteristics. This can be explained mainly by the differences in the workflow meta-models and by the longer development, improvements and evolution of workflow systems that took place in the field of enterprise application management (or the level of maturity reached by the workflow technology in this domain).

In recent years Cloud computing has gained significant acceptance in both the enterprise application management and scientific computing for its promise to reduce infrastructure costs and provide virtually unlimited computational power and data storage (Armbrust et al., 2009) — requirements of particular importance for businesses, and of even greater importance to scientists and research organizations. While research in this field is very active in providing novel concepts, techniques and principles towards building Cloud-native applications, there is a significant effort to Cloud-enable existing applications in order to reuse existing systems and therefore investments. Typically, Cloud-enabling applications is related to the migration of whole systems or parts of them on a public or private Cloud environment (Andrikopoulos et al., 2013), (Deelman et al., 2008). Current research in migration methodologies and techniques, both specific to the eScience domain and outside of it, is presented later in Section 1.3.

In this work we present a vendor- and technology-independent methodology for migrating the database layer of applications, and refactoring the application architecture as positioned in existing methodologies for migration of applications (see Section 1.4). The methodology is applicable to applications in different application domains and is agnostic to the types of data sources. It fulfills requirements also presented in this work, which we have identified in collaboration with software engineers and domain experts in several research projects. We use this methodology to migrate the database layer of a scientific workflow management system (SimTech SWfMS), which we developed in the scope of our research activities in the SimTech project. The architecture and implementation details of the system, as well as the motivation for the database layer migration, are first presented in Section 1.2. The migration of the SimTech SWfMS has been done using the Cloud Data Migration Support Tool — a proof of concept implementation of the methodology. Both the introduced methodology and the supporting tool have been evaluated and our findings are presented in Section 1.5. Our concluding remarks and plans for future work are presented in Section 1.6.

## 1.2 Motivating Scenario

As a motivating scenario from the eScience field we use the integrated and interactive *Scientific Workflow Management System* (SWfMS) developed in the context of the SimTech project (Sonntag et al., 2012), (Sonntag and Karastoyanova, 2010). The SimTech SWfMS is a distributed system based on conventional workflow technology adapted to the needs of scientific workflows. The main components of SimTech SWfMS are: a modeling and monitoring tool, a workflow engine, an enterprise service bus, an auditing system, a messaging system, several database management systems, and an application server running the simulation services.

We present the architecture of the SimTech SWfMS in Figure 1. The user interacts with the system using the modeling and monitoring tool. SimTech SWfMS provides a graphical user interface to model, execute, and monitor scientific workflows. When the user initiates the execution of a workflow, the tool automatically deploys the workflow model on the workflow engine which makes the simulation workflow available for use. The workflow can be instantiated as many times as needed. The instantiation of a scientific workflow is the beginning of the execution phase of the workflow life cycle.

The workflows executed by the workflow engine describe the ordered execution of different tasks like data preparation, computation or visualization. In our case these tasks are realized by Web services hosted on an application server. During the execution of a workflow the workflow engine navigates along the predefined control flow and also interacts with these Web services through the service bus, i.e. sends request for invocation of a Web service and receives the results of the back from the Web services. The service bus is also responsible for service discovery and selection if information about concrete services to be used is not available during the workflow deployment step. The workflow engine also produces fine granular workflow execution events and publishes them to the messaging system. These events are consumed by the modeling and monitoring tool as well as by the auditing application. The modeling and monitoring tool uses the execution information to enable the live monitoring of running workflows. The auditing application captures the same execution information and saves it into a database to enable provenance and later analysis.

The actual workflow which serves as an example in the following is a *Kinetic Monte Carlo Simulation* (KMS) which invokes several Web services as part of the simulation of solid bodies. These Web services are implemented by modules of the OPAL application (Sonntag et al., 2011). During their operation, the OPAL Web services access a MySQL database for both read and write operations. The example simulation of solid bodies is long-running and requires significant computing power. Speeding up the simulation was a challenge that led to the decision to make use of Cloud infrastructures, with

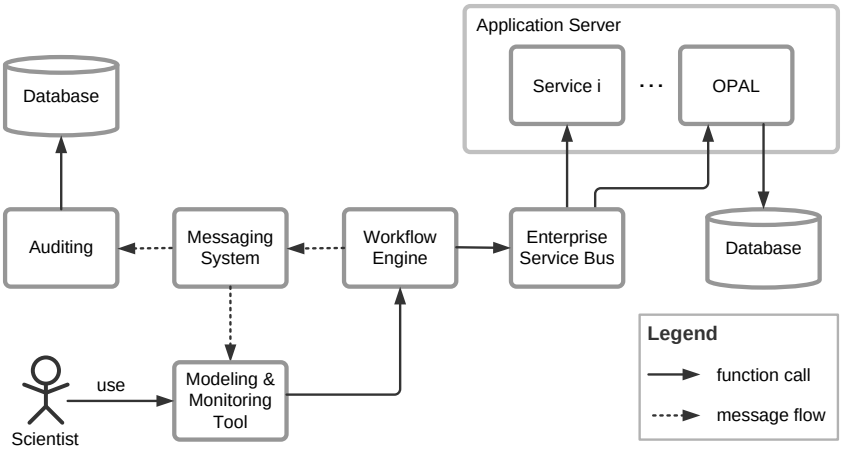



**Figure 1**
Main Components of the SimTech SWfMS Architecture.

the goal to acquire additional computational resources and data storage for the time of executing the simulation. This was indeed the major motivation for migrating the simulation workflow system or parts of it to the Cloud.

Since the example simulation produces big amounts of data one of our decisions was to temporarily migrate the database layer of the OPAL Web services into the Cloud, thus realizing the migration scenario *Cloud Bursting* (Strauch et al., 2013), with Amazon RDS as the migration target. Migrating however only the database layer to Amazon would result in extensive data transfer between the OPAL services on-premise and the database off-premise, and therefore creating a potential bottleneck. Considering this, we decided to migrate also the business logic of the application to the Cloud. Consequently, the modeling and monitoring tool was kept on-premise, while the remaining parts of the SimTech SWfMS were moved to an off-premise infrastructure. As a result, we do not only avoid bottlenecks, but in addition reduce costs, since for most Cloud providers data transfer inside the Cloud is significantly cheaper than data transfer from and to the Cloud.

The challenges we faced during this process were:

- which part of the system to migrate,
- what is the target system to migrate on,
- if and how to adapt the existing system to operate correctly after the migration,

- and most importantly, the lack of automated support with respect to the above decisions.

In order to address these challenges, in this work we present a methodology which incorporates decision and refactoring support for migration of the database layer of applications to the Cloud. For this purpose, in the following section we focus on investigating available methodologies and decision support systems for such scenarios.

## 1.3 Related Work

The State of the Art we investigate in this section covers three aspects. First, we review existing literature on recommendations, benefits, and use cases with respect to the usage of Cloud computing for eScience. Second, we investigate available vendor-specific and vendor-independent methodologies and guidelines for migrating either the database layer, or the whole application to the Cloud. Afterwards we consider available recommendation and decision support systems with respect to migration to the Cloud.

Mudge et al. reported a speedup by a factor of five on execution times when they migrated an eScience application from the domain of geophysics from on-premise to the Cloud, considering services from Amazon AWS and Microsoft Windows Azure (Mudge et al., 2011). Cala et al. used Cloud computing to satisfy the demand for increased computation power and need for storing large volumes of data by migrating an existing eScience application for predicting chemical activity to Microsoft Windows Azure (Cala et al., 2013). The migration scenarios we are using in our methodology do not only cover enterprise use cases but also scientific scenarios, as we have collaborated with industry partners and domain experts from eScience domain while identifying them. Zinn et al. migrated an existing application based on scientific workflows from the domain of astronomy to Microsoft Windows Azure (Zinn et al., 2010). The existing application we migrate to the Cloud for the purpose of evaluating our approach is also based on scientific workflows. Deelman et al. evaluated the cost of running eScience applications in the Cloud focusing on the trade-off between different workflow execution modes and provisioning plans, and came to the conclusion that the costs highly depend on the selected deployment strategy (Deelman et al., 2008). We do not explicitly consider costs, but provide recommendations and guidelines with respect to the deployment strategy.

Amazon proposes a phase-driven approach for migration of an application to their Cloud infrastructure consisting of six phases (Varia, 2010). The data migration phase is subdivided into a selection of the concrete Amazon AWS service and the actual migration of the data. Amazon provided recommendations regarding which of their data and storage services best fit for storing a

specific type of data, e.g. Amazon Simple Storage Service[3] is ideal for storing large write-once, read-many types of objects. As the methodology proposed by Amazon focuses on Amazon AWS data and storage services only, we abstract from this methodology and integrate the guidelines in our proposal. In addition to several product specific guidelines and recommendations (Microsoft, 2013a), (Microsoft, 2013b), Microsoft provides a Windows Azure SQL Database Migration Wizard[4] and the synchronization service Windows Azure SQL Data Sync[5]. We reuse some of these tools, tutorials and wizards and refer to them during the data migration phase.

Google is offering for the App Engine the tool Bulk Loader, which supports both the import of CSV and XML files into the App Engine Data Store and the export as CSV, XML, or text files[6]. The potentially required transformations of the data during the import are customizable in configuration files. In addition, Google supports the user when choosing the appropriate data store or service and during its configuration (Google, Inc., 2013b). Moreover, they provide guidelines to migrate the whole application to Google App Engine (Google, Inc., 2013a). We refer to the tools during the migration phase and abstract from the vendor-specific guidelines and recommendations in order to integrate them in our tool.

Salesforce provides data import support to their infrastructure via a Web UI or the desktop application Apex Data Loader[7]. Another option to migrate and integrate with Cloud providers such as Salesforce is to hire external companies that are specialized on migration and integration such as Informatica Cloud[8]. In addition to the tools or external support, Salesforce provides data migration guidelines (salesforce.com, Inc., 2013). We consider the non Salesforce-specific steps for our proposed methodology. As it will be discussed extensively in Sect. 1.4, Laszewski and Nauduri also propose a vendor-specific methodology for the migration to Oracle products and services by providing a detailed methodology, guidelines, and recommendations focusing on relational databases (Laszewski and Nauduri, 2011). We base our proposal on their methodology, by astracting from it, adapting and extending it.

Apart from the vendor specific migration methodologies and guidelines there are also proposals independent from a specific Cloud provider. Reddy and Kumar propose a methodology for data migration that consists of the following phases: design, extraction, cleansing, import, and verification. Moreover, they categorize data migration into storage migration, database migration, application migration, business process migration, and digital data retention (Reddy and Kumar, 2011). In our proposal we focus on the storage

[3]Amazon S3: `http://aws.amazon.com/s3/`

[4]Windows Azure SQL Migration Wizard: `http://sqlazuremw.codeplex.com`

[5]Windows Azure SQL Data Sync: `http://www.windowsazure.com/en-us/manage/services/sql-databases/getting-started-w-sql-data-sync/`

[6]Bulk Loader: `http://bulkloadersample.appspot.com`

[7]Apex Data Loader: `http://sforce-app-dl.sourceforge.net`

[8]Informatica Cloud: `http://www.informaticacloud.com`

and database migration as we address the database layer. Morris specifies four golden rules of data migration with the conclusion that the IT staff does not often know about the semantics of the data to be migrated, which causes a lot of overhead effort (Morris, 2012). With our proposal of a step-by-step methodology we provide detailed guidance and recommendation on both data migration and required application refactoring in order to minimize this overhead. Tran et al. adapted the function point method in order to estimate the costs of Cloud migration projects and classified the applications potentially migrated to the Cloud (Tran et al., 2011). As our assumption is that the decision to migrate to the Cloud has already been taken we do not consider aspects like costs. We abstract from the classification of applications in order to define the Cloud Data Migration Scenarios and reuse distinctions such as complete or partial migration in order to refine a chosen migration scenario.

As we provide the prototypical realization of a tool providing support and guidelines while deciding for a concrete Cloud data store or service, the migration, and the refactoring of the application architecture accordingly, in the following we also investigate the State of the Art on Decision Support Systems (DSS) (Power, 2002) in the area of Cloud computing. Khajeh-Hosseini et al. introduce two tools that support the user when migrating an application to IaaS Cloud services (Khajeh-Hosseini et al., 2011). The first one enables the cost estimation based on a UML deployment model of the application in the Cloud. The second tool helps to identify advantages and potential risks with respect to the Cloud migration. None of these tools is publicly available. We do not consider the estimation of costs, or the identification of risks as our assumption is that the decision for migration to the Cloud has already been taken. We consider aspects like costs, business resiliency, effort, etc. to be considered before following our methodology and using the tool (Andrikopoulos et al., 2013). Menzel et al. developed CloudGenius, a DSS for the selection of an IaaS Cloud provider focusing on the migration of Web servers to the Cloud based on virtualization technology (Menzel and Ranjan, 2012). As we provide support for the migration of the database layer we focus on another type of middleware technology. Our approach is also not limited to a specific Cloud service delivery model and migration by using virtualization technology.

## 1.4 Migration Methodology and Tool Support

As discussed above, in this section we introduce a step-by-step methodology for the migration of the database layer to the Cloud and the refactoring of the application architecture. Before we introduce the methodology, we investigate the requirements to be fulfilled by such a methodology.

### 1.4.1 Requirements

The *functional* and *non-functional* requirements we present in this section aim to provide decision support and guidelines for both migrating an application database layer to the Cloud, and for the refactoring of the application architecture. The presented requirements have been identified during our work in various research projects, and especially during our collaboration with industry partners and IT specialists from the eScience domain.

#### Functional Requirements

The following functional requirements must be fulfilled by any methodology for migration of the database layer to the Cloud and refactoring of the application architecture:

$FR_1$ *Support of Data Stores and Data Services*: The methodology must support the data migration for both fine- and coarse-grained types of interactions, e.g. through SQL and service APIs, respectively.

$FR_2$ *On-premise and Off-premise Support*: The methodology has to support data stores and data services that are either hosted on-premise or off-premise, and using both Cloud and non-Cloud technologies.

$FR_3$ *Independence from Database Technology*: The methodology has to support both established relational database management systems (Codd, 1970) and NoSQL data stores (Sadalage and Fowler, 2012) that have emerged in recent years.

$FR_4$ *Management and Configuration*: Any tool supporting such a methodology must provide management and configuration capabilities for data stores, data services, and migration projects bundling together different migration actions. This includes, for example, the registration of a new data store, including its configuration data, e.g. database schemas, database system endpoint URLs, etc. It must also support the creation of new migration projects for documentation of the decisions and actions taken during migration.

$FR_5$ *Support for Incompatibility Identification and Resolution*: Any potential incompatibilities, e.g. between SQL versions supported by different data services, must be identified, and guidance must be provided on how to overcome them. For this purpose the methodology has to incorporate the specification of functional and non-functional requirements for both the (source) database layer used before the migration, and for the target data store or data service.

$FR_6$ *Support for Various Migration Scenarios*: As the data migration depends on the context and the concrete use case, e.g. backup, archiving, or Cloud bursting, the methodology has to support various migration scenarios.

FR$_7$ *Support for Refactoring of the Application Architecture*: The amount of refactoring of the application architecture during the migration of the database layer to the Cloud depends on many aspects, such as the supported functionalities of the target data store or data service, use case, etc. It is therefore required that the methodology provides guidance and recommendations on how to refactor the application architecture.

**Non-functional Requirements**

In addition to the required functionalities, a methodology for migration of the database layer to the Cloud and refactoring of the application architecture should also respect the following properties:

NFR$_1$ *Security*: Both data export from a source data store, and data import to a target data store require confidential information such as data store location and access credentials. Any tool supporting the methodology should therefore consider necessary authorization, authentication, integrity, and confidentiality mechanisms and enforce user-wide security policies when required.

NFR$_2$ *Reusability*: As the migration of data can be either seen as the migration of only the database layer or as part of the migration of the whole application, the methodology has to be reusable with respect to the integration into a methodology for migration of the whole application to the Cloud, such as the one proposed by Varia for Amazon (Varia, 2010).

NFR$_3$ *Extensibility*: The methodology should be extensible to incorporate further aspects that impact the data migration to the Cloud, such as regulatory compliance. For example, in the US the Cloud service provider is responsible to ensure compliance to regulations (Louridas, 2010), but in the EU it is the Cloud customer that is ultimately responsible for investigating whether the provider realizes the Data Protection Directive (Cate, 1994).

### 1.4.2 Migration Methodology

The step-by-step methodology we introduce in this section refines and adapts the migration methodology proposed by Laszewski and Nauduri (Laszewski and Nauduri, 2011) in order to address the identified requirements.

The methodology in (Laszewski and Nauduri, 2011) consists of seven distinct phases (Fig. 2). During the *Assessment* phase, information relevant for project management such as drivers for migration, migration tools, and migration options is collected in order to assess the impact of the database migration on the IT ecosystem. The *Analysis and Design* phase investigates the implementation details on the target database, e.g. potentially different

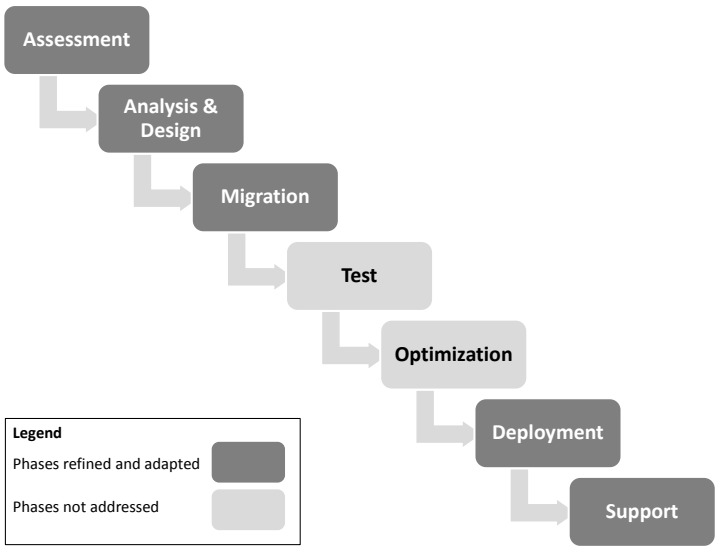



**Figure 2**
Migration Methodology as Proposed by (Laszewski and Nauduri, 2011), with Supported Phases Highlighted.

data types and transaction management mechanisms being used. The goal of this phase is the creation of a plan to overcome potential incompatibilities between the source and target data store, while avoiding changes in the business logic of the application. The *Migration* phase deals with the migration of the data from the source data store to the target data store in a testing environment, including tasks such as database schema migration, database stored procedures migration, and data migration. After the migration, both the database and the application have to be tested in the *Test* phase. This includes for example tasks such as data verification and testing the interaction of the application with the new target data store. As applications are in general highly optimized for a particular database, after the migration to another target data store the performance might be poor. Thus, optimizations based on the new target store used are applied in the *Optimization* phase in order to improve the performance. The goal of the *Deployment* phase is to deploy the final system, including actually migrating the database, to the production environment.

At first glance, the methodology of Laszewski and Nauduri addresses most of the requirements discussed in the previous. However, it discusses its phases on a high level that is not suitable for direct application, requiring further refinement in practice. Furthermore, it fails to satisfy some of the most important requirements that we identified. More specifically, as the methodology focuses on Oracle solutions it only considers the relational database man-

agement system of Oracle as target data store and the following relational data stores as source databases for the migration: Microsoft SQL Server[9], Sybase[10], IBM DB2[11], and IBM Informix[12]. All of these databases are data stores supporting fine-grained interactions through SQL. It is unclear whether the methodology also supports data services, as no information can be found on this aspect in (Laszewski and Nauduri, 2011) ($FR_1$). The methodology is not independent from the database technology as it focuses on a small set of relational databases and does not support NoSQL approaches ($FR_3$). Moreover, the methodology is limited to the pure outsourcing of the database layer to the Cloud and does not consider the context and specifics of migration scenarios such as Cloud bursting, backup, and archiving ($FR_6$). As concrete migration scenarios are not considered, their specifics and the context cannot be considered for the guidance and recommendation towards refactoring of the application architecture. In addition, the guidance and recommendations for the required adaptations of the application architecture during the migration are very limited, since the migration methodology in (Laszewski and Nauduri, 2011) considers only one vendor-specific relational target data store and a small subset of vendor-specific relational data stores as source data store ($FR_7$). The vendor-specificity has also the consequence that the methodology does not consider the reusability aspect with respect to the integration or combination of this methodology with other existing proposals for migration to the Cloud ($NFR_2$).

Addressing these deficiencies, in the following we propose a vendor- and database technology-independent step-by-step methodology which refines and adapts the one proposed in (Laszewski and Nauduri, 2011). Figure 2 provides an overview of the phases of the methodology proposed in (Laszewski and Nauduri, 2011) that we adapted and refined. Figure 3 provides an overview of our proposal consisting of seven steps. All steps are semi-automatic, in the sense that a human (e.g. the application developer in charge of the migration) has to provide input and follow the recommendations and guidelines provided by the methodology. Figure 3 also shows the mapping between the proposed methodology and the one in (Laszewski and Nauduri, 2011). As it can be seen, no direct support for the Test and Optimization phases is provided by our proposal since there are no identified requirements explicitly requiring these phases. The impact of not supporting these phases is evaluated in Section 1.5. The steps of the methodology are:

**Step 1: Select Migration Scenario**

The first step in our proposed methodology is the *selection of the migration scenario*. For this purpose, we use the ten *Cloud Data Migration Scenarios*

[9]Microsoft SQL Server: `http://www.microsoft.com/en-us/sqlserver`
[10]Sybase: `http://www.sybase.com`
[11]IBM DB2: `http://www.ibm.com/software/data/db2`
[12]IBM Informix: `http://www.ibm.com/software/data/informix/`

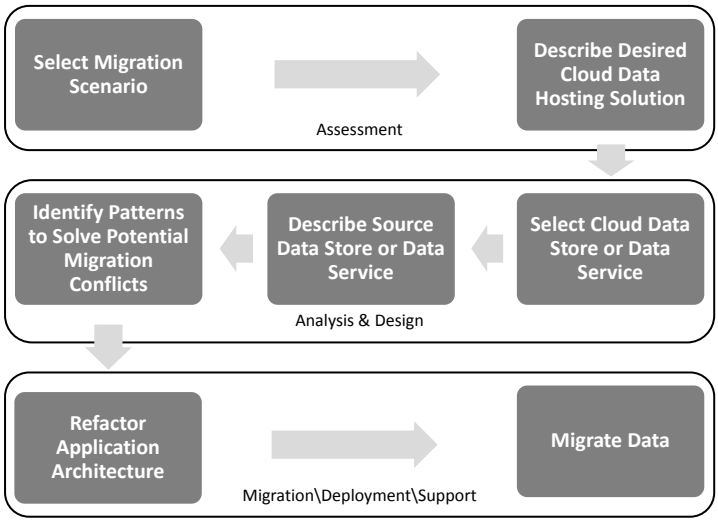



**Figure 3**
Methodology for Migration of the Database Layer to the Cloud and Refactoring of the Application Architecture.

identified in (Strauch et al., 2013): database layer outsourcing, using highly-scalable data stores, geographical replication, sharding, Cloud bursting, working on data copy, data synchronization, backup, archiving, and data import from the Cloud ($FR_6$). These migration scenarios cover both migration directions between on-premise and off-premise ($FR_2$).

Based on the selection of the migration scenario, a *migration strategy* is formulated by considering properties such as live or non-live migration, complete or partial migration, and permanent or temporary migration to the Cloud. During this step potential conflicts between the migration scenario selected and the refined migration strategy should be explicitly addressed by proposing solutions to the user, e.g. the choice of a different migration scenario. An example of a conflict is the selection of the migration scenario Cloud bursting and the choice of a permanent migration to the Cloud in the strategy. The purpose of this migration scenario is by definition to migrate the database layer to the Cloud in order to cover peak loads and migrate it back afterwards; choosing therefore permanent migration as part of the strategy cannot be satisfied.

### Step 2: Describe Desired Cloud Data Hosting Solution

The specification of functional and non-functional requirements with respect to the target data store or data service is the focus of the second step. We define *Cloud Data Hosting Solution* as the concrete configuration of a Cloud data store or Cloud data service in terms of a set of concrete functional and

non-functional properties ($FR_1$). Therefore, we derived an initial set of properties grouped into different categories based on the analysis of current data store and data service offerings of established Cloud providers such as Amazon, Google, and Microsoft. Table 1 provides an excerpt of the categories and corresponding properties we consider. These categories cover both relational and NoSQL solutions ($FR_3$, $FR_5$).

**Step 3: Select Cloud Data Store or Data Service**

The *concrete target data store or data service* for the migration is selected in step three by mapping the properties of the Cloud Data Hosting Solution specified in the previous step to the set of available data stores and data services that have been categorized according to the same non-functional and functional properties. Implementing this step requires data stores and data services to be previously specified according to the set of functional and non-functional properties either directly by the Cloud providers, or by the users of the methodology. The management and configuration capabilities required for this specification can however be used at a latter time to also make new Cloud data stores and data services available ($FR_4$).

**Step 4: Describe Source Data Store or Data Service**

As it is not sufficient to consider only where the data has to be migrated to, in step four the *functional and non-functional properties of the source data store or data service* are also described in order to identify and solve potential migration conflicts, e.g. the database technology used, or whether the location is on-premise or off-premise ($FR_5$).

**Step 5: Identify Patterns to Solve Potential Migration Conflicts**

The usage of Cloud technology leads to challenges such as incompatibilities with the database layer previously used or the accidental disclosing of critical data, e.g. by moving them to the Public Cloud. Incompatibilities in the database layer may refer to inconsistencies between the functionalities of an existing traditional database layer and the characteristics of an equivalent Cloud Data Hosting Solution. Therefore, in the fifth step conflicts are identified by checking the compatibility of the properties of the target data store selected in step three with the properties of the source data store or service used before the migration ($FR_5$). As a way to address these conflicts, in previous work (Strauch et al., 2013) we have defined a set of *Cloud Data Patterns* as the best practices to deal with them that can be reused here.

**Step 6: Refactor Application Architecture**

As the migration of the database layer also has an impact on the remaining application layers (presentation and business logic (Fowler et al., 2002)), the methodology should also provide guidelines and hints on what to be consid-

**Table 1**
Excerpt of Categories and Properties for Specification of Requirements of Cloud Data Hosting Solutions.

| Categories | Properties | Available Options |
|---|---|---|
| **Scalability** | Degree of Automation | Manual, Automated |
| | Type | Horizontal, Vertical |
| | Degree | Virtually Unlimited, Limited |
| | Time to Launch new Instance | None, Duration in Minutes |
| **Availability** | Replication | Yes, No |
| | Replication Type | Master-Slave, Master-Master |
| | Replication Method | Synchronous, Asynchronous |
| | Replication Location | Same Data Center, Different Data Center (Same Region) |
| | Automatic Failover | Yes, No |
| | Degree | 99.9%, 99.999% |
| **Security** | Storage Encryption | Yes, No |
| | Transfer Encryption | Yes, No |
| | Firewall | Yes, No |
| | Authentication | Yes, No |
| | Confidentiality | Yes, No |
| | Integrity | Yes, No |
| | Authorization | Yes, No |
| **Interoperability** | Data Portability | None, Import, Export, One-Way-Synchronization |
| | Data Exchange Format | XML, JSON, Proprietary |
| | Storage Access | SOA, REST-API, SQL, Proprietary |
| | ORM | JPA, JDO, LINQ |
| | Migration & Deployment Support | Yes, No |
| | Supported IDE | Eclipse, NetBeans, IntelliJ IDEA |
| | Developer SDKs | Java, .Net, PHP, Ruby |
| **Storage** | Storage Type | RDBMS, NoSQL |
| **CAP** | Consistency Model | Strong, Weak, Eventual |
| | Availability in Case of Partitioning | Available, Not Available |

ered for the refactoring of the application. Special focus should be given on the adaptation of the network, the data access layer, and the business logic layer of the application, depending on the outcomes of the previous steps ($FR_7$). Networking adaptation might require for example the reconfiguration of open ports in the enterprise firewall. Although the Cloud data store might be fully compatible with the data store previously used, the migration requires at least a change to the database connection string in the data access layer. The impact of the database layer migration to the Cloud on the business logic layer depends on several aspects, such as the migration scenario and the incompatibilities of the source and target data store. In case of switching from a relational database to a NoSQL data service, the business logic needs to be significantly adapted as the characteristics of these two technologies are different for example with respect to transaction support, relational database schema vs. schema-free or schema-less NoSQL solution, and Quality of Services (Sadalage and Fowler, 2012).

**Step 7: Migrate Data**

The final step, *migrating the data*, entails the configuration of the connections to the source and target data stores or services by requiring input on the location, credentials, etc. from the user. This step should also provide *adapters* for the corresponding source and target stores, bridging possible incompatibilities between them, and/or reuse of the data export and import tools offered by the different Cloud providers. As the last step is dealing with potentially confidential information, in order to prevent other users from accessing the data a tool supporting the proposed methodology has to support the required security mechanisms ($NFR_1$).

### 1.4.3 Realization

In this section we introduce the realization of a *Cloud Data Migration Tool* for the migration of the database layer to the Cloud and the refactoring of the application architecture (Strauch et al., 2013). More specifically, in order to support the proposed methodology, the Cloud Data Migration Tool provides two main functionalities. On the one hand it provides a repository for Cloud data stores and Cloud data services and allows browsing through it, even without user registration. Additionally, it implements the required management functionality to add new entries in the repository by specifying their functional and non-functional properties. On the other hand, the tool guides the user through the first six steps of the proposed methodology through a decision support system. For the last step of migrating the data, the tool is equipped with adapters that allow the automatic export of data from the source data store and their import in the target data store. Currently the

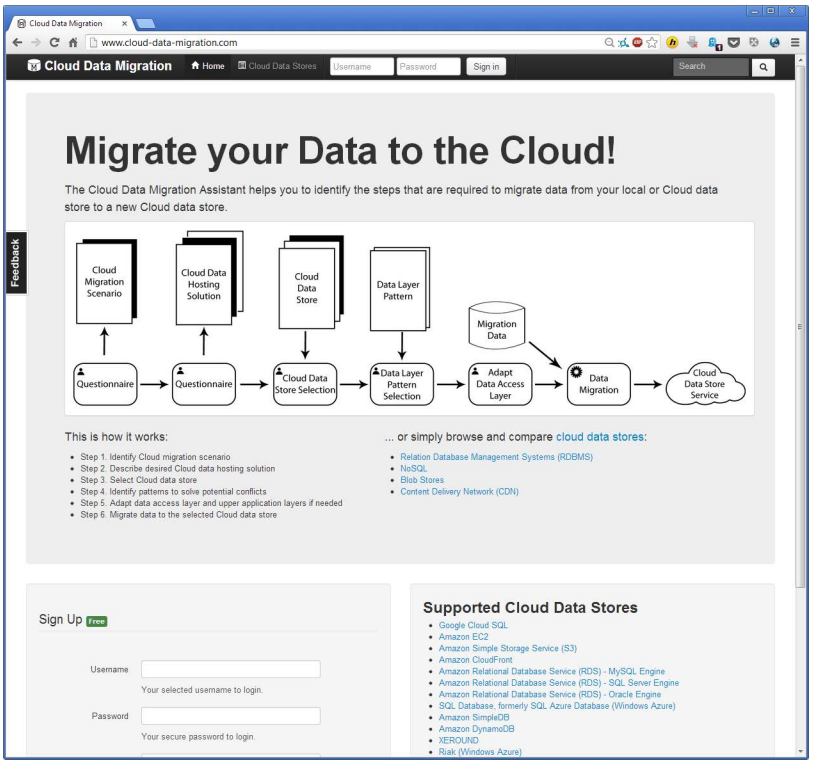



**Figure 4**
Screen Shot of the Realization of the Cloud Data Migration Tool.

tool has source adapters for PostgreSQL[13] and Oracle MySQL[14]. We provide target adapters for a number of Cloud Data Stores and Data Services like Amazon RDS[15] and 10gen MongoDB[16], MySQL in Amazon EC2 instances[17], Google Cloud SQL[18], and Amazon SimpleDB[19]. In addition to the adapters, the user is also referred to various guidelines and tutorials provided by the different Cloud providers, like e.g. (Google, Inc., 2013c). This is especially useful if no appropriate adapter is available for a particular data store or service.

Figure 4 provides an overview of the main page of the Cloud Data Migration Tool publicly available for free use[20]. As the user has to provide con-

[13] PostgreSQL: `http://www.postgresql.org`
[14] Oracle MySQL: `http://www.mysql.com`
[15] Amazon Relational Database Service: `http://aws.amazon.com/rds/`
[16] 10gen MongoDB: `http://www.mongodb.org`
[17] Amazon EC2: `http://aws.amazon.com/ec2/`
[18] Google Cloud SQL: `http://cloud.google.com/products/cloud-sql/`
[19] Amazon SimpleDB: `http://aws.amazon.com/simpledb/`
[20] Cloud Data Migration Tool: `http://www.cloud-data-migration.com`

fidential data following the guidelines and recommendations of the tool, e.g. access credentials to the source and target data stores or services for data export and import in the last step, he has to register with user, password, and e-mail address. After a migration project is finalized, the user can print a report of the decisions made during the migration, the identified conflicts and their resolutions for the purpose of documentation and support. Currently, we are supporting the migration from one source data store to one target data store or service and one migration project has to be created per migration. Extending the tool in order to support more than one target data stores per migration project is ongoing work.

The Cloud Data Migration Tool is realized as a Java 6 Web application and follows a three layer architecture. The presentation layer is realized using HTML, JavaScript, JSP, and CSS. The business logic layer is implemented in Java. For the object-relational mapping we use Java Data Objects version 3.1 and its implementation DataNucleus version 3.0[21]. For online hosting of the tool we use Google Cloud SQL as the data layer and run the whole application in Google's App Engine. A stand-alone, offline version of the tool also exists, allowing the user to run the tool locally. In this case MySQL 5.5 is used for the data layer and Apache Tomcat version 7 as the servlet container. Further information is available on the Web site of the Cloud Data Migration Tool `http://www.cloud-data-migration.com`.

## 1.5 Evaluation

In this section we evaluate both the methodology introduced in Section 1.4.2, and the Cloud Data Migration Tool supporting this methodology presented in the previous section. For this purpose we use the motivating scenario discussed in Section 1.2 as a case study involving the migration of the database layer of the SimTech SWfMS to the Cloud.

As our investigation of the literature did not result in a method that specifically aims at the evaluation of migration methodologies, we focused our analysis on related evaluation methods and standards for software processes and software quality. Al-Qutaish and Berander et al. provide an overview of available software quality models and standards (Al-Qutaish, 2010), (Berander et al., 2005). Based on their findings, we selected the ISO/IEC 9126 standard provided by International Organization for Standardization (ISO) and International Electrical Commission (IEC) for the evaluation of the Cloud Data Migration tool, as its quality attribute model includes the metrics we are considering as most relevant such as understandability and operability (Jung et al., 2004). For the evaluation of software processes there are multiple guide-

[21] DataNucleus: `http://www.datanucleus.org`

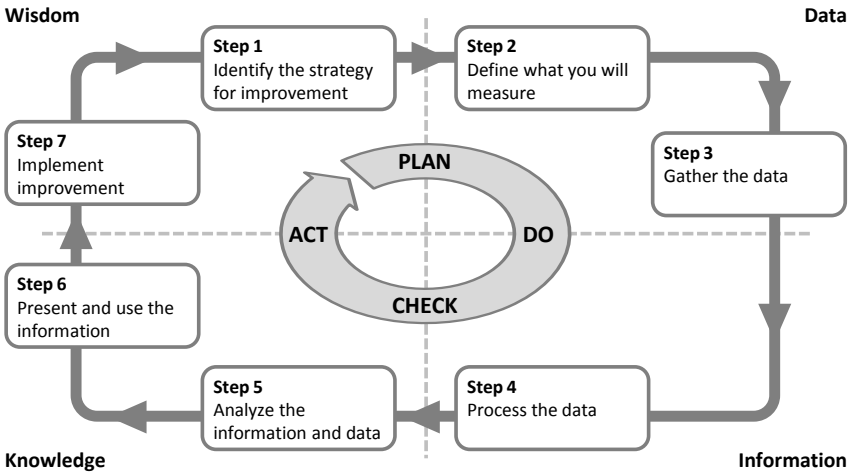



**Figure 5**
CSI Seven-Step Process Used for the Evaluation (Adapted from (Case and Spalding, 2011)).

lines, e.g. (Shull et al., 2001), (Sommerville, 1996), and standardized best practices such as Capability Maturity Model Integration (CMMI) (CMMI Product Team, 2010) and the Continual Service Improvement (CSI) module of the IT Infrastructure Library (ITIL) (Case and Spalding, 2011). We base our evaluation of the migration methodology on the ITIL CSI process, but adapt it in order to consider the technical aspects of the methodology by considering appropriate metrics for software processes provided by Daniel (Daniel, 2004). A simplified representation of the resulting process is shown in Fig. 5.

In the first step, a strategy for the realization of the process is determined. In this case, our strategy is to use the Cloud Data Migration Tool discussed in the previous section in conjunction with a specific migration scenario, and investigate whether it supports the scenario in an effective and efficient manner. In the second step, it needs to be defined which data will be collected. These data are the basis for the subsequent process steps. In our evaluation we collected both qualitative and quantitative data. With respect to the former, we recorded the user-identified problems that occured during the execution of the SimTech SWfMS migration as the means to evaluate the software quality of the Cloud Data Migration Tool. Such problems are gathered only in a qualitative manner, i.e. we are not interested in the number of occurred problems, but in a comprehensive description and classification of these problems. This approach increases the effort to gather the data, but in turn enables a more detailed and potentially more meaningful analysis. In terms of quantitative data, we recorded the time required for executing the various migration

phases. In order to be able to compare our proposal with the one by Laszewski and Nauduri (Laszewski and Nauduri, 2011), we chose to use the phases of the latter as the metric of the efficiency of our proposed approach. In this manner, we can attribute time elapsed to higher-level activities, in addition to evaluating the impact of not incorporating the testing and optimization phases in our proposal.

To enable a structured gathering and recording of occurring problems we have defined a set of attributes related to them. Table 2 shows an example of such a problem that was identified during our evaluation, and the information we collected for it. Every problem has a unique identifier (*ID*) and a descriptive *Name*. The attribute *Class* is used to classify the problem in predefined categories. We derived these categories from ISO/ICE 9126-1, which defines a quality model for software by subdividing software quality in different characteristics and sub characteristics (Jung et al., 2004). In our evaluation we focus on the characteristics *functionality* and *usability* of the examined tool, and in particular on the sub-characteristics *suitability* (for the former), and *understandability* and *operability* (for the latter), which are the possible values for the Class attribute. The problem identified in Table 2, for example, is classified under the operability sub-characteristic of usability. The attribute *Severity* describes the severity of a problem with respect to the impact on the migration result. The allowed values are *low*, *middle*, *high*, or *critical*. A detailed description of a problem is given with the attribute *Description*. The attribute *Error Handling* describes how the user has proceeded to find a solution for the occurred problem. *Solution* describes how the problem was fixed. To eliminate the cause of the problem, adaptations of the tool may be needed; these are described by the attribute *Adaptation*.

In the third step the actual gathering of data is performed. Using the Cloud Data Migration Tool, we migrated the database layer used by the OPAL Web services to the Cloud. The selected use case can be mapped to the migration scenario *Cloud Bursting* (Strauch et al., 2013), with Amazon RDS as the migration target. Throughout all phases of the migration we recorded any occurring problems, as shown in Table 2. In addition, we measured the time spent per migration phase supported by our step-by-step methodology (i.e. Assessment, Analysis & Design, and Migration, Deployment & Support), as well as the time spent on testing. No optimization activity was implemented as part of the case study. In the fourth step of the evaluation, the previously gathered data are processed in order to organize and structure for further analysis. As we have already gathered the data in a structured and uniform manner (as described in step 2), further processing is not necessary.

In the fifth step, the analysis of the gathered and processed data takes place. Altogether, we have recorded seven problems. Five of the recorded problems have a high priority; the remaining two have a middle priority. Two of the occurred problems are due to bugs in the graphical user interface of the tool, one with middle and one with high priority. Two problems were caused by missing features, also one with middle and one with high priority. The rest

**Table 2**
Documentation of an Identified Problem.

| | |
|---|---|
| **ID** | B7 |
| **Name** | Connection failed |
| **Class** | Tool (operability) |
| **Severity** | High |
| **Description** | Although correct users with the required administrative roles existed in the MySQL database in the Cloud, the application could not connect to the database. |
| **Error Handling** | We were going through all the security (user and privilege) settings in the MySQL Workbench. |
| **Solution** | We set *max queries, max updates, max connections* to a value greater than zero for each user. |
| **Adaptation** | The user should get information about the limitations for the different accounts (users). |

of the problems, all with high priority, were caused by lack of appropriate information available to the user, as in the example of Table 2. The analysis of the identified problems with respect to their priority and the cause of the problems shows that the main weakness of the Cloud Data Migration Tool is a lack of information provided to the user. Further improvements toward this direction are therefore required in the future.

The analysis of the time spent per migration phase is summarized Fig. 6. As shown in the figure, half of the time was actually spent in the Test phase, which as explained in Section 1.4 is not directly supported by our methodology (and therefore also not by the Cloud Data Migration Tool). While this identifies a deficiency in our proposal, it can also be attributed at least in part to the accelaration of the other phases by the use of the Cloud Data Migration Tool. In any case, what can be identified is a clear need for the incorporation of the remaining two phases (Test and Optimization) in our methodology, and as a result their support by the Cloud Data Migration Tool.

Finally, for the implementation of steps six and seven of the ITIL CSI process (presentation and use of the information, and implement improvements, respectively), we are currently in the process of incorporating the lessons learned by this case study in further research work.

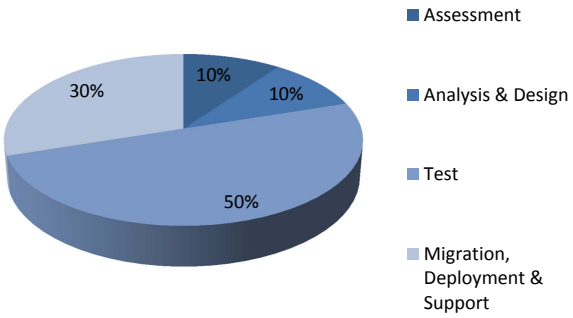



**Figure 6**
Amount of Time Spent per Migration Phase.

## 1.6 Conclusions

The popularity of Cloud computing has led to significant research in Cloud-enabling applications, i.e. migrating whole systems or only parts of them to the Cloud. The eScience domain, and especially the scientific workflow community, has reported concrete benefits from utilizing Cloud infrastructures for isolated use cases. In this respect, there is a clear need for a methodology supporting the migration of eScience applications to the Cloud. There are two key aspects that characterize eScience applications: large amounts of data, and intensive computational tasks to be performed on these data. In this work, we focus on the former, discussing how to support the migration of the database layer of eScience applications (and beyond) to the Cloud.

Supporting the migration of the database layer of an application to the Cloud involves not only considering the requirements on the appropriate data source or service imposed by the application, but also the possible need for adapting the application in order to cope with incompatibilities. In the previous sections we presented a step-by-step methodology that considers both aspects of the migration. In order to construct this methodology, we first identified a series of functional and non-functional requirements from both eScience and business domains. We then adapted the methodology discussed in (Laszewski and Nauduri, 2011) in order to satisfy the identified requirements, resulting in our proposal for a 7-step end-to-end methodology for the migration of the database layer to the Cloud and for the application refactoring required as part of this process.

Following on, we proceeded to discuss the realization of our proposal as a publicly available and free Cloud Data Migration Tool. The tool provides two fundamental functionalities: decision support in selecting an appropriate

data store or service, and refactoring support during the actual migration of the data. Users of the tool can currently create migration projects, define their requirements in terms of the migrated database layer to the Cloud, describe their current database layer and receive recommendations, hints and guidelines on where and how to migrate their data. Conflict resolution is based on previously identified Cloud Data Patterns, and data adapters are provided, allowing for the automatic migration of data to recommended data stores and services. We evaluated our proposal by migrating the SimTech Scientific Workflow Management System (SWfMS) to Amazon Web Services solutions, and showed that while useful, our methodology and tool need further improvements.

In particular, according to our evaluation, our proposal needs to be extended in order to provide explicit support for the testing phase of the migration. The Cloud Data Migration Tool must be extended to provide sandboxing capabilities, and both functional testing for bug fixing, and performance benchmarking tools for different application work loads. These capabilities can also be used towards supporting the optimization of the database layer after its migration. Additional functionalities that are currently being implemented to the Cloud Data Migration Tool, as identified in the previous sections, include addressing the impact of the migration to compliance, supporting more than one source and/or target data stores or services and multiple migrations per project, increasing the number of adapters available in the tool, as well as improving the usability of the tool for scientists.

## Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) projects 4CaaSt `http://www.4caast.eu` (grant agreement no. 258862) and ALLOW Ensembles `http://www.allow-ensembles.eu` (grant agreement no. 600792), and from the German Research Foundation (DFG) within the Cluster of Excellence in Simulation Technology `http://www.simtech.uni-stuttgart.de` (EXC 310/1) at the University of Stuttgart.

# *Bibliography*


Al-Qutaish, R. E. (2010). Quality Models in Software Engineering Literature: An Analytical and Comparative Study. *Journal of American Science 6*(3), 166–175.

Andrikopoulos, V., T. Binz, F. Leymann, and S. Strauch (2013). How to Adapt Applications for the Cloud Environment. *In: Computing, Springer 95(6)*, 493–535.

Armbrust, M. et al. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Berander, P. et al. (2005). Software Quality Attributes and Trade-Offs. Technical report, Blekinge Institute of Technology.

Cala, J., H. Hiden, S. Woodman, and P. Watson (2013). Cloud Computing for Fast Prediction of Chemical Activity. *Future Generation Computer Systems 29*(7), 1860–1869.

Case, G. and G. Spalding (2011). *ITIL Continual Service Improvement*. TSO, The Stationery Office.

Cate, F. (1994). The EU Data Protection Directive, Information Privacy, and the Public Interest. *Iowa L. Rev. 80*, 431.

CMMI Product Team (2010). CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). Software Engineering Institute, Carnegie Mellon University. `http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm`.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM 13*(6), 377–387.

Daniel, G. (2004). *Software Quality Assurance: From Theory To Implementation*. Pearson Education.

De Roure, D., C. Goble, and R. Stevens (2009). The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. *Future Generation Computer Systems 25*, 561–567.

Deelman, E., G. Singh, M. Livny, B. Berriman, and J. Good (2008). The Cost of Doing Science on the Cloud: the Montage Example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pp. 50:1–50:12. IEEE Press.

Fowler, M. et al. (2002, November). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

Google, Inc. (2013a). Google App Engine – Migrating to the High Replication Datastore. `http://developers.google.com/appengine/docs/adminconsole/migration`.

Google, Inc. (2013b). Google App Engine – Uploading and Downloading Data. `http://developers.google.com/appengine/docs/python/tools/uploadingdata?hl=en`.

Google, Inc. (2013c). Google Cloud SQL – Importing and Exporting Data. `http://developers.google.com/cloud-sql/docs/import\_export`.

Görlach, K., M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter (2011). Conventional Workflow Technology for Scientific Simulation. In *Guide to e-Science*, pp. 323–352. Springer.

Hey, A. J., S. Tansley, K. M. Tolle, et al. (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research Redmond, WA.

Jung, H.-W., S.-G. Kim, and C.-S. Chung (2004). Measuring Software Product Quality: A Survey of ISO/IEC 9126. *Software, IEEE 21*(5), 88–92.

Khajeh-Hosseini, A., I. Sommerville, J. Bogaerts, and P. Teregowda (2011). Decision Support Tools for Cloud Migration in the Enterprise. In *Proceedings of CLOUD'11*, pp. 541–548. IEEE.

Laszewski, T. and P. Nauduri (2011). *Migrating to the Cloud: Oracle Client/Server Modernization*. Elsevier.

Louridas, P. (2010). Up in the Air: Moving Your Applications to the Cloud. *Software, IEEE 27*(4), 6–11.

Menzel, M. and R. Ranjan (2012). CloudGenius: Decision Support for Web Server Cloud Migration. In *Proceedings of WWW'12*, pp. 979–988. ACM.

Microsoft (2013a). Develop and Deploy with Windows Azure SQL Database. `http://social.technet.microsoft.com/wiki/contents/articles/994.develop-and-deploy-with-windows-azure-sql-database.aspx`.

Microsoft (2013b). Guidelines and Limitations (Windows Azure SQL Database). `http://msdn.microsoft.com/en-us/library/windowsazure/ff394102.aspx`.

Morris, J. (2012). *Practical Data Migration* (2 ed.). BCS, The Chartered Institute for IT.

Mudge, J., P. Chandrasekhar, G. Heinson, and S. Thiel (2011). Evolving Inversion Methods in Geophysics with Cloud Computing - A Case Study of an eScience Collaboration. In *Proceedings of e-Science'11*, pp. 119–125.

Power, D. (2002). *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books.

Reddy, V. G. and G. S. Kumar (2011). Cloud Computing With a Data Migration. *Journal of Current Computer Science and Technology 1*(06).

Sadalage, P. J. and M. Fowler (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.

salesforce.com, Inc. (2013). Salesforce Help – Data Importing Overview. `http://help.salesforce.com/HTViewHelpDoc?id=importing.htm&language=en_US`.

Shull, F., J. Carver, and G. H. Travassos (2001). An Empirical Methodology for Introducing Software Processes. *SIGSOFT Softw. Eng. Notes 26*(5), 288–296.

Sommerville, I. (1996). Software Process Models. *ACM Comput. Surv. 28*(1), 269–271.

Sonntag, M., M. Hahn, and D. Karastoyanova (2012, September). Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In *Proceedings of CEUR Workshop'12*, pp. 1–5. Springer.

Sonntag, M., S. Hotta, D. Karastoyanova, D. Molnar, and S. Schmauder (2011). Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications. In *Towards a Service-Based Internet*, pp. 242–253. Springer.

Sonntag, M. and D. Karastoyanova (2010). Next Generation Interactive Scientific Experimenting Based on the Workflow Technology. In *Proceedings of MS'10*, pp. 349–356.

Strauch, S., V. Andrikopoulos, T. Bachmann, D. Karastoynova, S. Passow, and K. Vukojevic-Haupt (2013, December). Decision Support for the Migration of the Application Database Layer to the Cloud. In *Proceedings of CloudCom'13*. IEEE Computer Society Press.

Strauch, S., V. Andrikopoulos, T. Bachmann, and F. Leymann (2013). Migrating Application Data to the Cloud Using Cloud Data Patterns. In *Proceedings of CLOSER'13*, pp. 36–46. SciTePress.

Strauch, S., V. Andrikopoulos, U. Breitenbücher, S. G. Sáez, O. Kopp, and F. Leymann (2013). Using Patterns to Move the Application Data Layer to the Cloud. In *Proceedings of PATTERNS'13*, pp. 26–33. Xpert Publishing Services (XPS).

Taylor, I. J., E. Deelman, and D. B. Gannon (Eds.) (2006). *Workflows for e-Science: Scientific Workflows for Grids*. Springer.

Tran, V. T. K., K. Lee, A. Fekete, A. Liu, and J. Keung (2011). Size Estimation of Cloud Migration Projects With Cloud Migration Point (CMP). In *Proceedings of ESEM'11*, pp. 265–274. IEEE.

Varia, J. (2010). Migrating your Existing Applications to the AWS Cloud. A Phase-driven Approach to Cloud Migration.

Zinn, D., Q. Hart, B. Ludascher, and Y. Simmhan (2010). Streaming Satellite Data to Cloud Workflows for on-demand Computing of Environmental Data Products. In *Proceedings of WORKS'10*, pp. 1–8.