



## Decision Support for the Migration of the Application Database Layer to the Cloud

Steve Strauch, Vasilios Andrikopoulos, Thomas Bachmann, Dimka Karastoyanova, Stephan Passow, and Karolina Vukojevic-Haupt

Institute of Architecture of Application Systems,  
University of Stuttgart, Germany  
{lastname}@iaas.uni-stuttgart.de

---

BIB<sub>T</sub>E<sub>X</sub>:

```
@inproceedings{StrauchABKPV2013,  
  author    = {Steve Strauch, Vasilios Andrikopoulos, Thomas Bachmann, Dimka  
              Karastoyanova, Stephan Passow, Karolina Vukojevic-Haupt},  
  title     = {Decision Support for the Migration of the Application Database  
              Layer to the Cloud},  
  booktitle = {Proceedings of the 5th IEEE International Conference on Cloud  
              Computing Technology and Science, CloudCom 2013,  
              2-5 December 2013, Bristol, UK},  
  year      = {2013},  
  pages     = {639--646},  
  publisher = {IEEE Computer Society},  
  doi       = {10.1109/CloudCom.2013.90}  
}
```

© 2013 IEEE Computer Society. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.



## Decision Support for the Migration of the Application Database Layer to the Cloud

Steve Strauch, Vasilios Andrikopoulos, Thomas Bachmann, Dimka Karastoyanova, Stephan Passow, Karolina Vukojevic-Haupt  
*Institute of Architecture of Application Systems (IAAS),  
 University of Stuttgart, Stuttgart, Germany  
 {firstname.lastname}@iaas.uni-stuttgart.de*

**Abstract**—Migrating an existing application to the Cloud is a complex and multi-dimensional problem requiring in many cases adapting the application in significant ways. Looking specifically into the database layer of the application, i.e. the aspect providing data persistence and manipulation capabilities, this involves dealing with differences in the granularity of interactions, refactoring of the application to cope with remote data sources, and addressing data confidentiality concerns. Toward this goal, in this work we present an application migration methodology which incorporates these aspects, and a decision support, application refactoring and data migration tool that assists application developers in realizing this methodology. For purposes of evaluating our proposal we present the results of a case study conducted in the context of an eScience project.

**Keywords**—Data Migration, Decision Support, Database layer, Application Refactoring

### I. INTRODUCTION

The popularity of Cloud computing due to the promised reduction of capital expenses and the virtually infinite resource capacity [1] has motivated many application developers to *Cloud-enable* their applications, i.e. to migrate them to the Cloud. Cloud service providers like Amazon Web Services report a plethora of successful migrations of applications to their services<sup>1</sup>. Furthermore, the introduction of new Platform and Software as a Service (PaaS and SaaS, respectively) solutions complements successfully the already dominant Infrastructure as a Service (IaaS) model. Together, these service delivery models offer multiple options in the migration of one or more functional components of an existing application to potentially multiple service providers, resulting in different types of migration [2]. Supporting the decision on whether and how to migrate an application becomes in this manner a multi-dimensional problem with multiple decisions that need to be made [3].

In order therefore to scope the discussion, we focus this work on the migration of the *database layer* of the application, i.e. the data persistence and data manipulation aspect of the application [4]. In this context, and depending on the intention, direction and temporality of the migration, different scenarios can be realized. In [5], for example, the authors identify ten distinct migration scenarios ranging from complete outsourcing of the data to a Cloud hosting solution,

to temporal replication of the local database in the Cloud to cope with increased demand (Cloud bursting).

Data hosting solutions may also differ significantly with respect to the interaction of the application with the data store [2]. There can be either fine grained interactions using e.g. SQL to execute operations on a migrated database, or coarse grained interactions through pre-defined service APIs. Furthermore, as discussed extensively in [2], migration may result in the need to adapt the application, even in the case of “simply” moving a database to a VM in the Cloud. Beyond re-wiring of the application to access remote data, significant changes to the application logic may be required due to incompatibilities and performance concerns. In addition to potential application refactoring, data confidentiality needs to be considered when data are moved to the Cloud.

Migrating the database layer of an application to the Cloud requires therefore to consider a number of aspects when selecting which (public) Cloud data hosting solution to use. Additional support has also to be provided to the application developers in the case that application adaptations are unavoidable for the preferred solution. Our fundamental assumption in this case is that the decision to migrate the database layer to the Cloud has already been taken. Thus, we do not consider aspects like pricing and business resiliency [2].

The contributions of this work addressing these concerns can be summarized by the following:

- An application migration methodology geared towards the database layer;
- A decision support and migration tool that supports users in realizing this methodology; and
- An evaluation of the proposed methodology and tool based on a case study performed in the context of an eScience project.

The remainder of this paper is structured as follows: Section II discusses the requirements for the migration of a database layer to the Cloud, and our proposal for a step-by-step methodology that addresses them. Section III presents the Cloud Data Migration Tool, a decision support and application refactoring tool that is used for the realization of this methodology. The proposed methodology and tool are used for evaluation purposes during the migration of an existing eScience-related system to the Cloud. The structure

<sup>1</sup>AWS Case Studies: <https://aws.amazon.com/solutions/case-studies/>

and results of this evaluation are presented in Section IV. Related work is summarized in Section V. The paper concludes in Section VI, discussing also future work.

## II. METHODOLOGY

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.

### A. Requirements

The presented *functional* and *non-functional* 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. In particular:

*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<sub>1</sub> *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<sub>2</sub> *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<sub>3</sub> *Independence from Database Technology:* The methodology has to support both established relational database management systems [6] and NoSQL data stores [7] that have emerged in recent years.
- FR<sub>4</sub> *Management and Configuration:* Any tool supporting such a methodology must provide management and configuration capabilities for data stores, data services, and migration projects for the documentation of the decisions and actions taken during migration. This includes, for example, the registration of a new data store, including its configuration data, e.g. database schemas, database system endpoint URLs, etc.
- FR<sub>5</sub> *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.
- FR<sub>6</sub> *Support for Various Migration Scenarios:* As the data migration depends on the context and the concrete use case, the methodology has to support various migration scenarios, e.g. outsourcing, backup, archiving.
- FR<sub>7</sub> *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<sub>1</sub> *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<sub>2</sub> *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 [8].
- NFR<sub>3</sub> *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, but in the EU it is the Cloud customer that is ultimately responsible for investigating whether the provider realizes the Data Protection Directive [9].

### B. Migration Methodology

The step-by-step methodology we introduce in this section refines and adapts the migration methodology proposed by Laszewski and Nauduri [10] in order to address the identified requirements. The methodology in [10] consists of seven distinct phases (Fig. 1): in *Assessment*, information relevant for project management such as migration tools and migration options is collected in order to assess the impact of the database migration. *Analysis and Design* investigates the implementation details on the target database, e.g. potentially different data types and transaction management mechanisms being used, and creates a plan to overcome such incompatibilities. *Migration* deals with the migration of the data from the source data store to the target data store in a testing environment. After migration, both the database and the application have to be tested in the *Test* phase; this includes tasks such as data verification. Any necessary optimizations based on the new target store are applied during *Optimization*. Finally, 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 identified requirements. However, it

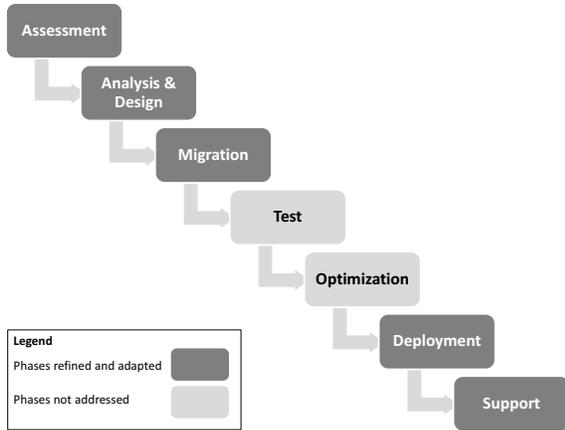


Figure 1. Migration Methodology as proposed by [10], with supported phases highlighted

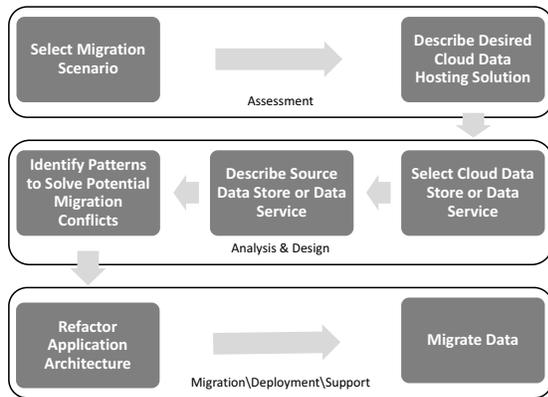


Figure 2. Methodology for migration of the database layer to the Cloud and refactoring of the application architecture

discusses its phases on a very high level, requiring further refinements in practice. Furthermore, it fails to satisfy some of the most important requirements that we identified. For example, it only supports Oracle solutions as target data stores (FR<sub>1</sub>), no NoSQL support is provided (FR<sub>3</sub>), and it is limited to the pure outsourcing of the database layer to the Cloud scenario (FR<sub>6</sub>).

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 [10]. Figure 2 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 2 also shows the mapping between the proposed methodology and the one in [10]. 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 IV.

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* identified in [5]: 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<sub>6</sub>). These migration scenarios cover both migration directions between on-premise and off-premise (FR<sub>2</sub>). 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.

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, *Describe Desired Cloud Data Hosting Solution*. 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<sub>1</sub>). Examples for such properties are transaction support, e.g. ACID characteristics of the data store, and the consistency model supported, e.g. strong or eventual consistency [11]. These categories cover both relational and NoSQL solutions (FR<sub>3</sub>, FR<sub>5</sub>).

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 can also be used at a later time to include new Cloud data stores and data services (FR<sub>4</sub>).

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<sub>5</sub>).

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<sub>5</sub>). As a way to address these conflicts we apply a set of *Cloud Data Patterns*, defined in our previous work [12], as the best practices to deal with them.

Since the migration of the database layer also has an impact on the remaining application layers (presentation and business logic [4]), the methodology should also provide guidelines and hints on what to be considered 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<sub>7</sub>). 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 [7].

The final step, *migrating the data*, entails the configuration of the connections to the source and target data stores or services by requiring from the user information about the location, credentials, etc. 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. Since this step deals with potentially confidential information, in order to prevent other users from accessing these data, a supporting tool has to guarantee the required security properties (NFR<sub>1</sub>).

### III. 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. More specifically, in order to support the proposed methodology, the Cloud Data Migration Tool

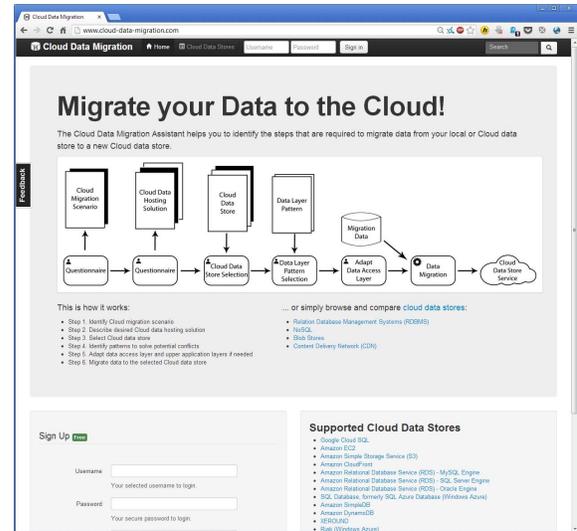


Figure 3. Screen shot of the realization of the Cloud Data Migration Tool

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 tool has source adapters for PostgreSQL<sup>2</sup> and Oracle MySQL<sup>3</sup>. We provide target adapters for a number of Cloud Data Stores and Data Services like Amazon RDS<sup>4</sup> and 10gen MongoDB<sup>5</sup>. In addition to the adapters, the user is also referred to various guidelines and tutorials provided by the different Cloud providers, like e.g. [13]. This is especially useful if no appropriate adapter is available for a particular data store or service.

Figure 3 provides an overview of the main page of the Cloud Data Migration Tool publicly available for free use<sup>6</sup>. As the user has to provide confidential 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

<sup>2</sup>PostgreSQL: <http://www.postgresql.org>

<sup>3</sup>Oracle MySQL: <http://www.mysql.com>

<sup>4</sup>Amazon Relational Database Service: <http://aws.amazon.com/rds/>

<sup>5</sup>10gen MongoDB: <http://www.mongodb.org>

<sup>6</sup>Cloud Data Migration Tool: <http://www.cloud-data-migration.com>

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<sup>7</sup>. 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>.

#### IV. EVALUATION

For purposes of evaluating the proposed methodology and the Cloud Data Migration Tool, we conducted a case study in the eScience field using an integrated and interactive Scientific Workflow Management System (SWfMS) developed in the context of the SimTech project [14], [15]. This SWfMS is based on conventional workflow technology adapted to the needs of scientific workflows and implemented as a distributed system. Its main components are: a modeling and monitoring tool, a workflow engine, an auditing system, a messaging system, several database management systems, and an application server running simulation services. The actual workflow, which serves as an example for our case study, models a *Kinetic Monte Carlo Simulation* (KMS) for solid bodies by orchestrating several Web services. These Web services are implemented by modules of the OPAL application [16]. During their operation, the OPAL Web services access a MySQL database which is to be migrated to the Cloud.

More specifically, the case study itself consists of the temporary migration of the database layer of the OPAL Web services to the Cloud. The selected use case can therefore be mapped to the migration scenario *Cloud Bursting* [5], with Amazon RDS as the migration target. To properly execute and document our case study, we used the seven-step improvement process that is typically followed by the *Continual Service Improvement (CSI)* module of the *IT Infrastructure Library (ITIL)* [17]. The goal of this process is to identify weaknesses of IT services and to derive possible improvements. A simplified representation of this process is shown in Fig. 4.

<sup>7</sup>DataNucleus: <http://www.datanucleus.org>

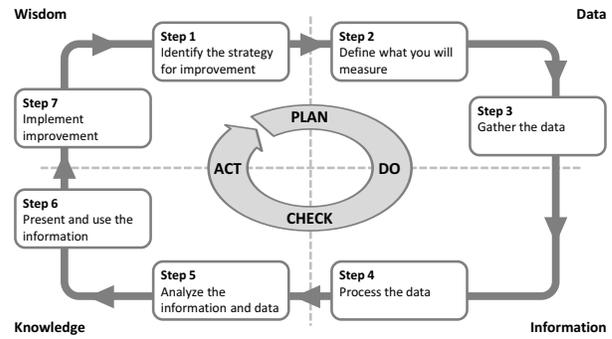


Figure 4. CSI seven-step process used for the evaluation (adapted from [17])

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, the data to be collected needs to be defined. 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 occurred 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 errors, but in a comprehensive description and classification of these problems. This approach increases the effort to gather the data, but on the other hand 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 in [10], we chose to use the phases of the latter as the metric of the efficiency of our 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 I 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 [18]. 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 *understand-*

Table I  
DOCUMENTATION OF AN IDENTIFIED PROBLEM

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

ability and operability (for the latter). The problem identified in Table I, for example, is classified under the operability sub-characteristic of usability. The attribute *Severity* describes the severity of the problem impact on the migration result. The allowed values are *low*, *middle*, *high*, or *critical*. The attribute *Description* stands for a detailed description of a problem. *Error Handling* describes how the user has proceeded to find a solution for the occurred problem and *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 (i.e. the MySQL DB) to the Cloud (i.e. Amazon RDS). Throughout all phases of the migration we recorded any occurring problems, following the example in Table I. 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, 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 of the problems, all with high priority, were caused by lack of appropriate information available to the user, as in the example of Table I. The analysis of the identified problems with respect to their

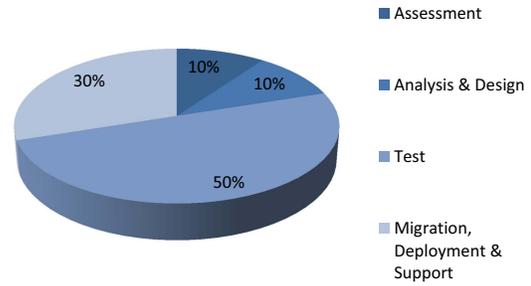


Figure 5. Quantitative data: amount of time spent per migration phase

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. 5, which shows that half of the time was actually spent in the Test phase, which, as explained in Section II, 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 acceleration of the other phases by the use of the Cloud Data Migration Tool. In any case, what can be identified is a clear need to incorporate the remaining two phases (Test and Optimization) in our methodology, and consequently their support in 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.

## V. RELATED WORK

With respect to vendor-specific methodologies for application migration, Amazon proposes a phase-driven approach for migration of an application to their Cloud infrastructure consisting of six phases [8]. The data migration phase is subdivided into a selection of the concrete Amazon AWS service and the actual migration of the data. Amazon also provides recommendations regarding which of their data and storage services best fit for storing a specific type of data, e.g. Amazon Simple Storage Service<sup>8</sup> 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. Furthermore, as discussed in Section II, Laszewski and Nauduri also propose a vendor-specific methodology for the migration

<sup>8</sup>Amazon S3: <http://aws.amazon.com/s3/>

to Oracle products and services by providing a detailed methodology, guidelines, and recommendations focusing on relational databases [10]. We base our proposal on their methodology, by refining and extending it.

In addition to several product specific guidelines and recommendations [19], [20], Microsoft provides a Windows Azure SQL Database Migration Wizard<sup>9</sup> and the synchronization service Windows Azure SQL Data Sync<sup>10</sup>. Google is offering 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<sup>11</sup>. 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 [21]. Moreover, they provide guidelines to migrate the whole application to Google App Engine [22]. Salesforce provides data import support to their infrastructure via a Web UI or the desktop application Apex Data Loader<sup>12</sup>. 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<sup>13</sup>. In addition to the tools or external support, Salesforce provides data migration guidelines [23]. All these guidelines were considered for the proposed methodology, and links to many of the offered tools are incorporated as recommendations in the Cloud Data Migration Tool.

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 [24]. In our proposal we focus on the storage 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 [25]. With our proposed step-by-step methodology, we provide detailed support for both data migration and required application refactoring in order to minimize this overhead.

In the area of Decision Support Systems (DSS) for Cloud computing, Khajeh-Hosseini et al. introduce two tools that support the user when migrating an application to IaaS Cloud services [26]. 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. We do neither consider the estimation of costs nor 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 [2]. Besides, none of the tools is publicly available. 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 [27]. As we provide support for the migration of the database layer and refactoring of the application we focus on another type of middleware technology. Our approach is thus not limited to a specific Cloud service delivery model and migration by using virtualization technology.

## VI. OUTLOOK AND FUTURE WORK

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 this work we presented a step-by-step methodology that considers these two aspects of migration. In order to construct this methodology, we first identified a series of functional and non-functional requirements. We then adapted the methodology discussed in [10] in order to satisfy the identified requirements, resulting in 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.

We also presented 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 on the target data store or service, 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 provided for the automatic migration of data to recommended data stores and services. We evaluated our proposal by migrating a scientific workflow management system to the Cloud. We 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 migration testing phase. The Cloud Data Migration Tool must be extended to provide sandboxing capabilities and both functional testing for bug fixing as well as performance

<sup>9</sup>Windows Azure SQL Migration Wizard: <http://sqlazuremw.codeplex.com/>

<sup>10</sup>Windows Azure SQL Data Sync: <http://www.windowsazure.com/en-us/manage/services/sql-databases/getting-started-w-sql-data-sync/>

<sup>11</sup>Bulk Loader: <http://bulkloadersample.appspot.com>

<sup>12</sup>Apex Data Loader: <http://sforce-app-dl.sourceforge.net>

<sup>13</sup>Informatica Cloud: <http://www.informaticacloud.com/>

benchmarking tools for different application work loads. These capabilities can also be used to support the optimization of the database layer after its migration. Additional functionalities that are currently being implemented in the Cloud Data Migration Tool include addressing the impact of the migration to compliance, security and data confidentiality, supporting more than one source and/or target data stores or services and multiple migrations per project, as well as increasing the number of adapters available in the tool.

#### ACKNOWLEDGMENTS

The research leading to these results has received funding from the EU's Seventh Framework Programme (FP7/2007-2013) projects 4CaaSt (grant agreement no. 258862) and ALLOW Ensembles (grant agreement no. 600792), and from the German Research Foundation (DFG) within the Cluster of Excellence in Simulation Technology.

#### REFERENCES

- [1] M. Armbrust *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.
- [2] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to Adapt Applications for the Cloud Environment," *In: Computing, Springer*, vol. 95(6), pp. 493–535, 2013.
- [3] V. Andrikopoulos, S. Strauch, and F. Leymann, "Decision Support for Application Migration to the Cloud," in *Proceedings of CLOSER'13*. SciTePress, May 2013, pp. 149–155.
- [4] M. Fowler *et al.*, *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, November 2002.
- [5] S. Strauch, V. Andrikopoulos, T. Bachmann, and F. Leymann, "Migrating Application Data to the Cloud Using Cloud Data Patterns," in *Proceedings of CLOSER'13*. SciTePress, 2013, pp. 36–46.
- [6] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.
- [7] P. J. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley, 2012.
- [8] J. Varia, "Migrating your Existing Applications to the AWS Cloud. A Phase-driven Approach to Cloud Migration," 2010.
- [9] P. Louridas, "Up in the Air: Moving Your Applications to the Cloud," *Software, IEEE*, vol. 27, no. 4, pp. 6–11, 2010.
- [10] T. Laszewski and P. Nauduri, *Migrating to the Cloud: Oracle Client/Server Modernization*. Elsevier, 2011.
- [11] W. Vogels, "Eventually Consistent," *Commun. ACM*, vol. 52, no. 1, pp. 40–44, 2009.
- [12] S. Strauch, V. Andrikopoulos, U. Breitenbücher, S. Gómez Sáez, O. Kopp, and F. Leymann, "Using Patterns to Move the Application Data Layer to the Cloud," in *Proceedings of PATTERNS'13*. Xpert Publishing Services (XPS), 2013, pp. 26–33.
- [13] Google, Inc., "Google Cloud SQL – Importing and Exporting Data," [http://developers.google.com/cloud-sql/docs/import\\_export](http://developers.google.com/cloud-sql/docs/import_export).
- [14] M. Sonntag and D. Karastoyanova, "Next Generation Interactive Scientific Experimenting Based on the Workflow Technology," in *Proceedings of MS'10*, 2010, pp. 349–356.
- [15] M. Sonntag, M. Hahn, and D. Karastoyanova, "Mayflower - Explorative Modeling of Scientific Workflows with BPEL," in *Proceedings of CEUR Workshop'12*. Springer, September 2012, pp. 1–5.
- [16] M. Sonntag, S. Hotta, D. Karastoyanova, D. Molnar, and S. Schmauder, "Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications," in *Towards a Service-Based Internet*. Springer, 2011, pp. 242–253.
- [17] G. Case and G. Spalding, *ITIL Continual Service Improvement*. TSO, The Stationery Office, 2011.
- [18] H.-W. Jung, S.-G. Kim, and C.-S. Chung, "Measuring Software Product Quality: A Survey of ISO/IEC 9126," *Software, IEEE*, vol. 21, no. 5, pp. 88–92, 2004.
- [19] Microsoft, "Guidelines and Limitations (Windows Azure SQL Database)," <http://msdn.microsoft.com/en-us/library/windowsazure/ff394102.aspx>.
- [20] —, "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>.
- [21] Google, Inc., "Google App Engine – Uploading and Downloading Data," <http://developers.google.com/appengine/docs/python/tools/uploadingdata?hl=en>.
- [22] —, "Google App Engine – Migrating to the High Replication Datastore," <http://developers.google.com/appengine/docs/adminconsole/migration>.
- [23] salesforce.com, Inc., "Salesforce Help – Data Importing Overview," [http://help.salesforce.com/HTViewHelpDoc?id=importing.htm&language=en\\_US](http://help.salesforce.com/HTViewHelpDoc?id=importing.htm&language=en_US).
- [24] V. G. Reddy and G. S. Kumar, "Cloud Computing With a Data Migration," *Journal of Current Computer Science and Technology*, vol. 1, no. 06, 2011.
- [25] J. Morris, *Practical Data Migration*, 2nd ed. BCS, The Chartered Institute for IT, 2012.
- [26] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda, "Decision Support Tools for Cloud Migration in the Enterprise," in *Proceedings of CLOUD'11*. IEEE, 2011, pp. 541–548.
- [27] M. Menzel and R. Ranjan, "CloudGenius: Decision Support for Web Server Cloud Migration," in *Proceedings of WWW'12*. ACM, 2012, pp. 979–988.

All links were last followed on September 19, 2013.