Institute of Architecture of Application Systems

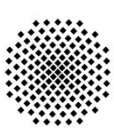
Migrating Enterprise Applications to the Cloud: Methodology and Evaluation

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

Nikolay Nachev and Albrecht Stäbler
NovaTec Holding GmbH,
Leinfelden-Echterdingen, Germany
{firstname.lastname}@novatec-gmbh.de

BibTeX:

```
@ARTICLE{,
author = {Strauch, Steve and Andrikopoulos, Vasilios and Karastoynova, Dimka
          and Leymann, Frank and Nachev, Nikolay and Staebler, Albrecht},
title = {Migrating Enterprise Applications to the Cloud: Methodology and
         Evaluation},
journal = {International Journal of Big Data Intelligence},
publisher = {Perpetual Innovation Media Pvt. Ltd},
pages = {127--140},
volume = {1},
number = {3},
url = {www.inderscience.com/ijbdi},
year = {2014}
}
```

University of Stuttgart
Germany

Migrating enterprise applications to the cloud: methodology and evaluation

Steve Strauch*, Vasilios Andrikopoulos, Dimka Karastoyanova and Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart,
Universitaetsstrasse 38,
70569 Stuttgart, Germany
Email: steve.strauch@iaas.uni-stuttgart.de
Email: vasilios.andrikopoulos@iaas.uni-stuttgart.de
Email: dimka.karastoyanova@iaas.uni-stuttgart.de
Email: frank.leymann@iaas.uni-stuttgart.de
*Corresponding author

Nikolay Nachev and Albrecht Stäbler

NovaTec Holding GmbH,
Dieselstrasse 18/1,
70771 Leinfelden-Echterdingen, Germany
Email: nikolay.nachev@novatec-gmbh.de
Email: albrecht.staebler@novatec-gmbh.de

Abstract: Migrating existing on-premise applications to the cloud is a complex and multi-dimensional task and may require adapting the applications themselves significantly. For example, when considering the migration of the database layer of an application, which provides data persistence and manipulation capabilities, it is necessary to address aspects like differences in the granularity of interactions and data confidentiality, and to enable the interaction of the application with remote data sources. In this work, we present a methodology for application migration to the cloud that takes these aspects into account. In addition, we also introduce a tool for decision support, application refactoring and data migration that assists application developers in realising this methodology. We evaluate the proposed methodology and enabling tool using a case study in collaboration with an IT enterprise.

Keywords: data migration; application migration; decision support; database layer; application refactoring; cloud computing.

Reference to this paper should be made as follows: Strauch, S., Andrikopoulos, V., Karastoyanova, D., Leymann, F., Nachev, N. and Stäbler, A. (2014) 'Migrating enterprise applications to the cloud: methodology and evaluation', *Int. J. Big Data Intelligence*, Vol. 1, No. 3, pp.127–140.

Biographical notes: Steve Strauch works as a Research Associate and PhD student at the Institute of Architecture of Application Systems (IAAS) at the University of Stuttgart since 2008. His research interests are data migration, data hosting, as well as data security and privacy in the area of cloud computing, with an emphasis on their architectural aspects. He has contributed to the European projects COMPAS, 4CaaSt, ALLOW Ensembles, and the German Government funded BMBF project ECHO.

Vasilios Andrikopoulos is a Senior Researcher at IAAS, University of Stuttgart. His research is in the areas of services science, cloud computing and infrastructures, and software engineering with an emphasis on evolution and adaptation. He received his PhD in 2010 from Tilburg University, the Netherlands, where he was also a member of the European Research Institute in Service Science (ERISS). He has experience in research and teaching database systems and management, software modelling and programming, business process management and integration, service engineering and cloud computing. He has participated in a number of EU projects, including NoE S-Cube and 4CaaSt.

Dimka Karastoyanova is a Junior Professor at the Institute of Architecture of Application Systems (IAAS) and in the Cluster of Excellence 'Simulation Technology' at the University of Stuttgart. She has received her Doctoral in Computer Science from TU Darmstadt, Germany. Her research interests and teaching activities include service-oriented computing and architecture, service middleware, business process management, flexible service compositions and adaptive systems, semantics and security aspects of service-based applications.

Frank Leymann is a Full Professor of Computer Science and Director of the Institute of Architecture of Application Systems (IAAS) at the University of Stuttgart, Germany. His research interests include service-oriented architectures and associated middleware, workflow- and business process management, cloud computing and associated systems management aspects, and patterns. The projects he is working on are funded by the European Union, the German Government, or directly by industry partners. He is the co-author of about 300 peer reviewed papers, more than 40 patents, and several industry standards (e.g., BPEL, BPMN, TOSCA). He is an invited expert to consult the European Commission in the area of cloud computing. Before accepting the professor position at University of Stuttgart, he worked for two decades as an IBM Distinguished Engineer where he was a member of a small team that was in charge of the architecture of IBM's complete middleware stack.

Nikolay Nachev works as a Consultant at NovaTec GmbH since October 2013. His interests are in the area of server administration, cloud computing and object-oriented programming. He has contributed to the projects CIMS (code.google.com/p/stuproa-cims) and moby (http://moby.iao.fraunhofer.de).

Albrecht Stäbler is the Chairman of the Board of the NovaTec GmbH, and is holding a university degree as Diplom-Ingenieur and is teaching software engineering, architectures and operating system software, middleware and workflow management at several universities in Germany and abroad. In strategic customer projects, he provides his experiences and his know-how in the field of large-scale software architecture and enterprise architecture as well as doing project audits and crisis management. As CEO, he is responsible for the corporate strategy. In the field of cloud engineering, he is the advisor for the cloud-provisioning product automaIT (http://www.automait.de/1/home/) and therefore is doing research and applied research in the fields cloud native architectures, TOSCA and others.

This paper is a revised and expanded version of a paper entitled 'Decision support for the migration of the application database layer to the cloud' presented at the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom'2013), Bristol, UK, 2–5 December 2013.

1 Introduction

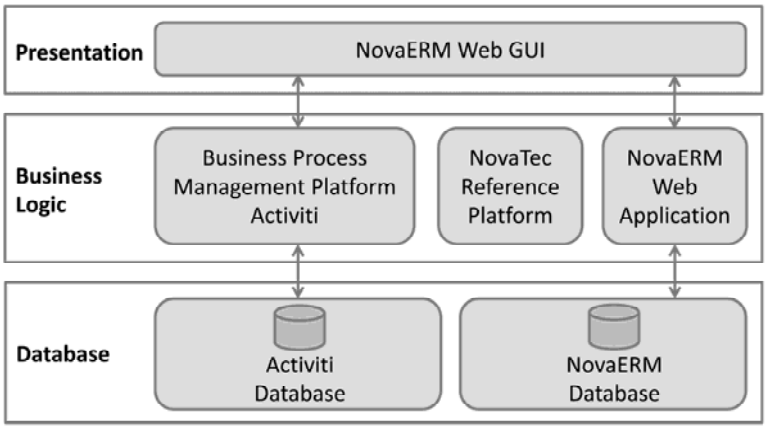
For its promise to reduce infrastructure costs and provide virtually unlimited computational power and data storage, as Armbrust et al. (2009) discuss, in recent years, cloud computing has gained significant acceptance in both the enterprise application management and scientific computing. While active research in this field provides novel concepts, techniques and principles towards building cloud-native applications, there is a significant effort, led by enterprises, to cloud-enable existing applications in order to reuse existing systems and therefore investments. Typically, as postulated by Andrikopoulos et al. (2013), cloud-enabling applications are related to the migration of whole systems or parts of them on a public or private cloud environment. More details on current research in migration methodologies and techniques are presented in Section 3.

In this work, we present a vendor- and technology-independent methodology for migrating the database layer of applications and refactoring the application, and position it in existing application migration methodologies (see Section 4). The methodology is applicable to applications in different application domains and is agnostic to the types of data sources. The requirements this methodology meets have been identified in collaboration with software engineers and domain experts in several research projects and collaborations with enterprises. For the evaluation of our approach, we use the NovaERM application developed in the scope of a company internal IT project at NovaTec Holding GmbH[1] (in the following referred to as NovaTec). We use this methodology for a partial and a complete migration of NovaERM. The architecture and implementation details of the system, as well as the motivation for migration, are presented in Section 2. The migration of the NovaERM application has been done using our cloud data migration support tool. The evaluation of the methodology and tool, and our findings are presented in Section 5. Our concluding remarks and plans for future work are in Section 6.

2 Motivating scenario

As a motivating scenario from the enterprise field we use the integrated and interactive *enterprise resource management application NovaERM* developed in Java in the context of a company-internal project at NovaTec. NovaERM was developed in order to automate and support company-internal business processes such as the hiring process, which we use as an example in the following. As we migrate NovaERM in the scope of our evaluation to the cloud, we present the system architecture of NovaERM in Figure 1.

Figure 1 Overview of NovaERM system architecture



The user interacts with the application using the *NovaERM Web GUI* which provides a graphical user interface to login, complete manual human tasks, enter data, and administer the system. The business logic layer contains the following three components: the *business process management* (*BPM*) *Platform Activiti*, the *NovaTec Reference Platform*, and the *NovaERM Web Application*. All company-internal business processes such as the hiring process are deployed and executed within the *Activiti BPM Platform version 5.12.1*.[2] The *NovaTec Reference Platform* builds the basis for the *NovaERM Web Application* by providing non-application-specific functionality. The *NovaERM Web Application* consists of several sub-components representing a partner, e.g., an employee or job candidate, and a contract. All sub-components are realised using services. The NovaERM Web GUI and NovaERM Web Application are running within a GlassFish application server[3] version 3.2.1. Finally, the database layer consists of the *Activiti Database* and the *NovaERM Database* using PostgreSQL[4] version 9.1.9. The Activiti Database stores the data generated by the Activiti BPM Platform while the processes are being deployed and executed. The NovaERM Database contains all the data relevant for the hiring process such as contract details and administration information for the whole system.

Considering the expansion of NovaTec, the company decided to migrate some of their operations to the cloud. In particular, it was needed to decide what is the solution with the least impact on their current architecture. Thus, in collaboration with NovaTec, we created a field study to evaluate their options.

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 (DSSs) for such scenarios.

3 Related work

First, 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 DSSs with respect to migration to the cloud.

In Varia (2010), Amazon proposes a phase-driven approach for migration of an application to their cloud infrastructure consisting of the following six phases: *cloud assessment*, *proof of concept*, *data migration*, *application migration*, *leverage the cloud*, and *optimisation*. The data migration phase is subdivided into a selection of the concrete Amazon AWS service and the actual migration of the data. We use this methodology by applying the first four phases and refined and implemented the *data migration* phase by using our proposed methodology for the migration of the database layer in order to evaluate the possibility to integrate our proposal into a methodology to migrate the whole application to the cloud. Additionally, 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[5] is ideal for storing large write-once, read-many types of objects. As the methodology proposed by Amazon focuses on AmazonAWSdata 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, 2013b), Microsoft provides a Windows Azure SQL Database Migration Wizard[6] and the synchronisation service Windows Azure SQL Data Sync.[7] 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.[8] The potentially required transformations of the data during the import are customisable in configuration files. In addition, Google, Inc. (2013b) supports the user when choosing the appropriate data store or service and during its configuration. Moreover, Google, Inc. (2013a) provides guidelines to migrate the whole application to Google App Engine. 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.[9] Another option to migrate and integrate with cloud providers such as Salesforce is to hire external companies that are specialised on migration and integration such as Informatica Cloud.[10] In addition to the tools or external support, salesforce.com, Inc. (2013) provides data migration guidelines. We consider the non-Salesforce-specific steps for our proposed methodology. As it will be discussed extensively in Section 4, Laszewski and Nauduri (2011) 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. We base our proposal on their methodology, by abstracting 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. Jamshidi et al. (2013) identified, taxonomically classified, and systematically compared existing research on cloud migration. We considered the lessons learned for the methodology, we propose and addressed the identified lack of tool support for enhancing cloud migration by implementing a tool realising our proposed methodology.

Reddy and Kumar (2011) propose a methodology for data migration that consists of the following phases: design, extraction, cleansing, import, and verification. Moreover, they categorise data migration into storage migration, database migration, application migration, business process migration, and digital data retention. In our proposal, we focus on the storage and database migration as we address the database layer. Morris (2012) 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. 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 minimise this overhead. Tran et al. (2011) adapted the function point method in order to estimate the costs of cloud migration projects and classified the applications potentially migrated to the cloud. 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 realisation 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 DSSs as defined in Power (2002) in the area of cloud computing. Khajeh-Hosseini et al. (2011) introduce two tools that support the user when migrating an application to IaaS cloud services. 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 as discussed in Andrikopoulos et al. (2013). Menzel and Ranjan (2012) developed CloudGenius, a DSS for the selection of an IaaS cloud provider focusing on the migration of web servers to the cloud based on virtualisation technology. 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 virtualisation technology.

Menychtas et al. (2013) investigate a model-driven approach for the migration of legacy applications to the cloud and present an integrated framework supporting this approach. Based on the fact that there is not always a model for the legacy system to be migrated and that periodically changing requirements for example with respect to scalability imply periodic updates of the model, we do not follow a model-driven approach for migration and refactoring of the application architecture.

Leymann et al. (2011) propose a method based on application model enrichment and a corresponding tool chain that allows moving an application to the cloud. In comparison to their approach, we introduce a DSS guiding the user through a step-by-step methodology without the need for an application model.

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.

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 enterprise domain.

4.1.1 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 as discussed by Codd (1970) and NoSQL data stores as discussed by 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.

4.1.2 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 authorisation, 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 (2010) for Amazon.

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 USA, the cloud service provider is responsible to ensure compliance to regulations as discussed by Louridas (2010), but in the EU it is the cloud customer that is ultimately responsible for investigating whether the provider realises the Data Protection Directive as stated in Cate (1994).

4.2 Migration methodology

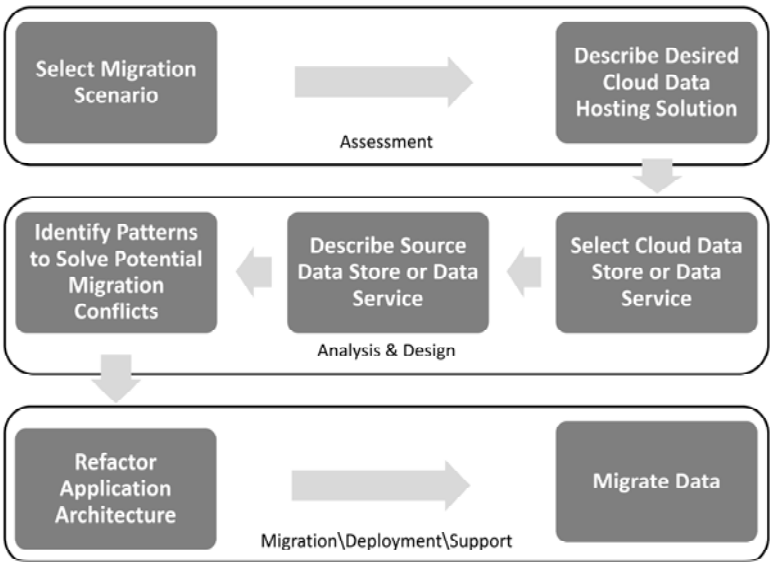
The step-by-step methodology, we introduce in this section refines and adapts the migration methodology proposed in Laszewski and Nauduri (2011) in order to address the identified requirements. The methodology in Laszewski and Nauduri (2011) consists of seven distinct phases. During the *assessment* 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 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 optimised for a particular database, after the migration to another target data store the performance might be poor. Thus, optimisations based on the new target store used are applied in the *optimisation* 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 (2011) 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 management system of Oracle as target data store and the following relational data stores as source databases for the migration: Microsoft SQL Server[11], Sybase[12], IBM DB2[13], and IBM Informix[14]. 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 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 Laszewski and Nauduri (2011). As it can be seen, no direct support for the test and optimisation 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 5. The steps of the methodology are.

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



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* identified in Strauch et al. (2013a): database layer outsourcing, using highly-scalable data stores, geographical replication, sharding, cloud bursting, working on data copy, data synchronisation, 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).

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 centre, different data centre (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
	Authorisation	Yes, no
Interoperability	Data portability	None, import, export, one-way-synchronisation
	Data exchange format	XML, JSON, proprietary
	Storage access	SOA, REST-API, SQL, proprietary
	ORM	JPA, JDO, LINQ
	Migration and 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

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 categorised 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. (2013c) 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 as described in Fowler et al. (2002)], 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_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 o f 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 (see 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).

4.3 Realisation

In this section, we introduce the realisation 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 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 DSS. 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[15] and Oracle MySQL.[16] We provide target adapters for a number of cloud data stores and data services like Amazon RDS[17] and 10gen MongoDB[18], MySQL in Amazon EC2 instances[19], Google Cloud SQL[20], and Amazon SimpleDB.[21] 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 3 Screen shot of the realisation of the cloud data migration tool (see online version for colours)

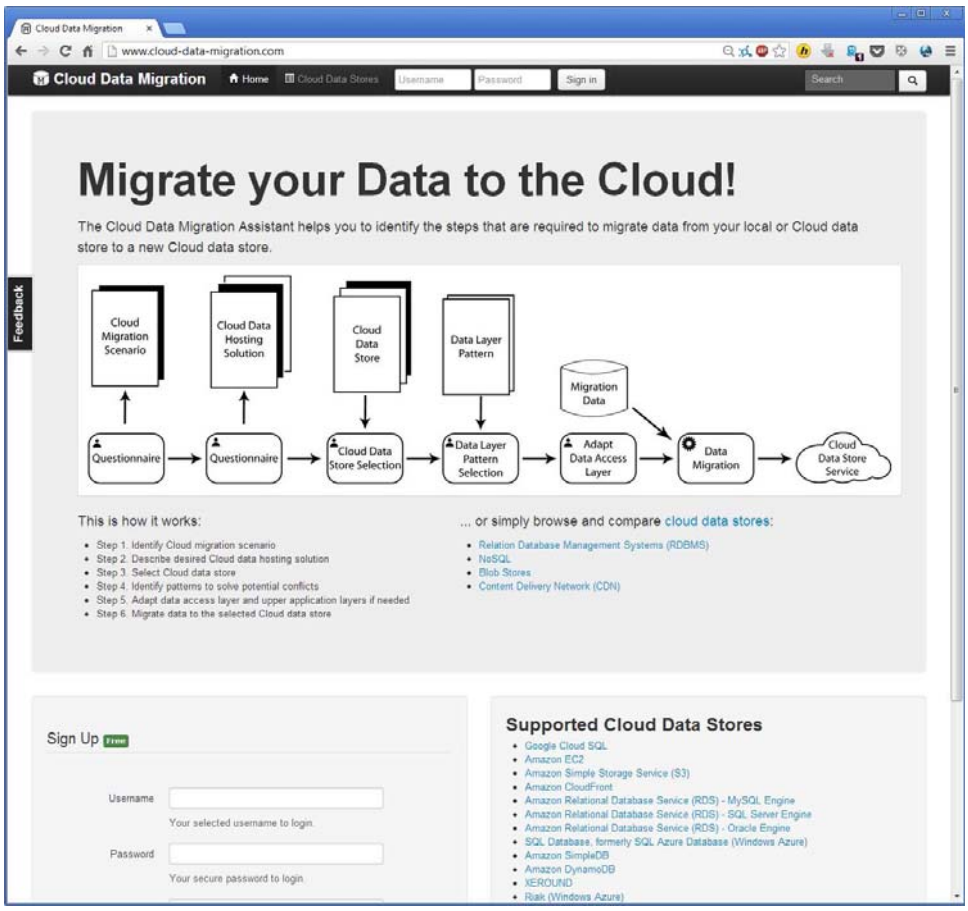


Figure 3 provides an overview of the main page of the cloud data migration tool publicly available for free use.[22] As the user has to provide confidential data following the guidelines and recommendations of the tool, e.g., access credentials to the source and target data stores or services for data export and import in the last step, he has to register with user, password, and e-mail address. After a migration project is finalised, 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 realised as a Java 6 web application and follows a three layer architecture. The presentation layer is realised 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.[23] 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 in Strauch et al. (2013b) and on the website of the cloud data migration tool http://www.cloud-data-migration.com.

5 Evaluation

In this section, we evaluate both the methodology introduced in Section 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 2 as a field study, as defined by Taylor-Powell and Steele (1996), involving both the partial migration of the NovaERM application by migrating the database layer only, and the complete migration of NovaERM to the cloud [Type II and Type III in the classification of Andrikopoulos et al. (2013)].

5.1 Method

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. For the evaluation of software processes there are multiple guidelines, e.g., Shull et al. (2001) and Sommerville (1996), and standardised best practices such as the capability maturity model integration

(CMMI) from the CMMI Product Team (2010) and the continual service improvement (CSI) module of the IT Infrastructure Library (ITIL) by 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 Kan (2002). The goal of this process is to identify weaknesses of IT services and to derive possible improvements. A simplified representation of the resulting process is shown in Figure 4.

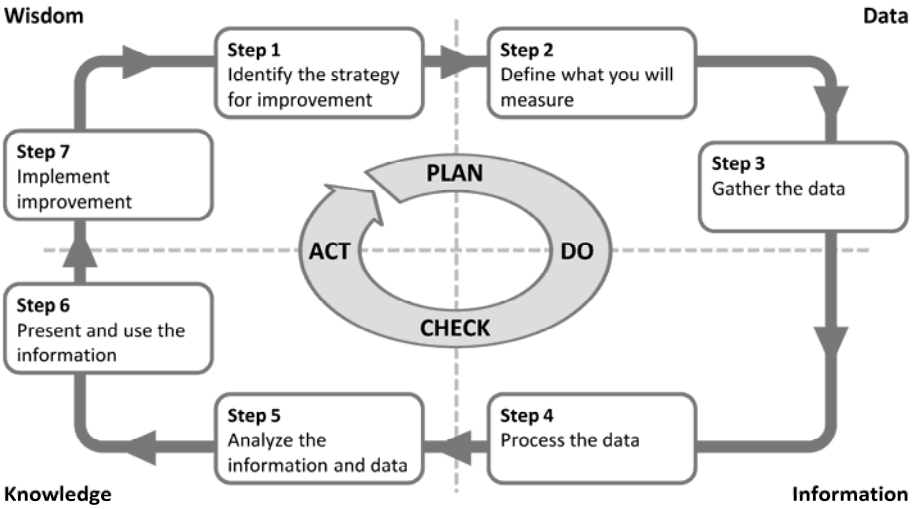
Berander et al. (2005) and Al-Qutaish (2010) provide an overview of available software quality models and standards. Based on their findings, we selected the ISO/IEC 25010 *Software product Quality Requirements and Evaluation (SQuaRE)* standard provided by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) for the evaluation of the cloud data migration tool. The quality in use model and system/software product quality model of SQuaRE includes the metrics we are considering as most relevant. For the quality in use, we focus on *efficiency and effectiveness* and for product quality we consider *functional suitability*, i.e., functional completeness and functional correctness, and *usability*, i.e., learnability and appropriateness ISO/IEC (2005). In order to evaluate these metrics, we recorded the user-identified problems that occurred during the execution of the partial and complete migration of NovaERM to the cloud as the means to evaluate the software quality of the cloud data migration tool. Such problems were 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.

5.2 Evaluation setup

In order to consider both the partial migration of NovaERM by migrating the database layer only, and the complete migration to the Cloud, we split the evaluation into two iterations. In the first iteration, we migrate the NovaERM Database and the Activiti Database to the NovaTec-internal Private Cloud and keep the other components of NovaERM locally, which is Migration Type II in Andrikopoulos et al. (2013) classification. During the second iteration, we migrate the whole software stack of NovaERM to the Amazon Public Cloud (Migration Type III). In order to evaluate the reusability of our proposed methodology with respect to the integration into a methodology for the migration of the whole application, we use the methodology proposed by Varia (2010) for Amazon and integrated our proposed methodology by using it as refinement and implementation of the data migration phase. In both iterations, we use virtual machines (VMs) to host NovaERM partially or completely in the cloud. Table 2 provides the specification of the VMs used.

In order to speed up the setup and configuration of the components of NovaERM both at the initial, local topology and during the migration to the cloud we used the provisioning solution automaIT[24] version 1.2, a commercial product of NovaTec which has been proven in various industry projects. In order to ensure that the NovaERM application works correctly after provisioning it partially and completely in the cloud through regression tests, we used the software testing tool selenium HQ[25] version 2.34 which enables browser automation, e.g., for automating web applications for testing purposes. Thirty-four test cases based on the hiring process were provided by the NovaTec internal development team for this purpose.

Figure 4 CSI seven-step process used for the evaluation



Source: Adapted from Case and Spalding (2011)

Table 2 Properties of VMs used

Properties	*NovaTec private cloud*	*Amazon EC2*
Instance Size	n/a	m1.medium
Vendor ID	GenuineIntel	GenuineIntel
CPU model name	Intel Xeon CPU E5-2640@2.5GHz	Intel Xeon CPU E5-2650@2.0GHz
CPU MHz	2500.000	1795.672
CPU cache size	15360 KB	20480 KB
Total Memory	3833 MB	3750 MB
Distribution ID	CentOS	Ubuntu
Distribution Release	6.4	12.04.2 LTS
AMI-ID	n/a	ami-ddfae2a9

Figure 5 Overview of duration of the database layer migration during first and second iteration (see online version for colours)

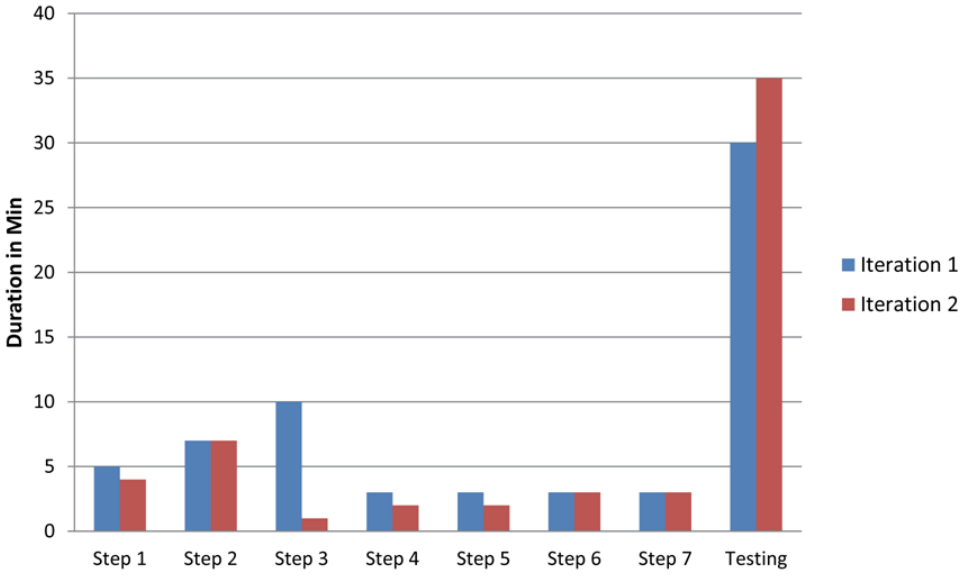


Table 3 Overview of duration of the Amazon methodology integrated with the proposed methodology

Phase	*Duration in minutes*
Cloud assessment	10
Proof of concept	1,440
Data migration	32
Application migration phase	20
Testing	35

5.3 Evaluation results

In this section, we present the evaluation results and discuss the lessons learned. Figure 5 provides an overview of the duration for each of the steps for the migration of the database layer to the cloud executed by a domain expert in the first and second iteration. No optimisation activity was implemented as part of the field study. Compared to the first iteration, the total duration of the migration of the database layer decreased in the second iteration by 12 minutes (measuring only the migration of the database). Thus, we conclude that there appears to be a shallow learning curve in using the cloud data migration tool, which in addition to the reusability of the data entered during the first iteration, are the main reasons for the faster migration results.

Table 3 presents the duration of the phases when migrating NovaERM completely to Amazon using the proposed methodology integrated into the migration methodology by Varia (2010) by refining and implementing it in the second iteration. The results show that there is an increase of the duration of the data migration phase of ten minutes compared to the duration of the proposed methodology as shown in Figure 5, because the migration to the Public Cloud of Amazon required further security configurations, in addition to network latency despite using the EUWest (Ireland) region of AWS. During the second iteration, we discovered that the proposed methodology

does not only cover the data migration phase, but also impacts the cloud assessment phase and application migration phase of the methodology from Amazon by accelerating them. From the amount of time spent on testing during both iterations and on the proof of concept phase in the second iteration, we conclude the need of incorporating support for testing and optimisation into our methodology and tool in the future.

In order to enable a structured gathering and recording of occurring problems we have defined a set of attributes related to them. Table 4 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 derived from the ISO/IEC 25010 according to the focus of our evaluation ISO/IEC (2005). With respect to the quality in use we consider *efficiency* and *effectiveness* and for product quality we focus on functional suitability, i.e., *functional completeness* and *functional correctness*, and usability, i.e., *learnability* and *appropriateness*, which are the possible values for the Class attribute. The problem identified in Table 4, for example, is classified under the functional suitability sub-characteristic of functional correctness and under the usability sub-characteristic appropriateness. 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/adaptation* describes how the problem was fixed or how to eliminate the cause of the problem by adaptations of the tool that may be required.

Altogether, we have recorded seven problems. One of the recorded problems has a critical priority (see Table 4) the remaining six have a middle priority. Five of the occurred problems are due to bugs in the graphical user interface or the business logic of the tool, one with critical and four with middle priority. The rest of the problems were caused by missing features, e.g., the domain expert requested a migration status information during the actual data migration. 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 are bugs that need to be fixed and a lack of missing features requested by the domain expert in order to improve the functional suitability and usability. Finally, for the implementation of the improvements (step seven of the ITIL CSI process, see Figure 4), we are currently in the process of incorporating the lessons learned by this field study in further research work.

6 Conclusions

Enterprises have reported concrete benefits from utilising cloud infrastructures for isolated use cases. The growing popularity of cloud computing has led to significant research in cloud-enabling applications, in particular with respect to migrating whole systems or only parts of them to the cloud. In this respect, there is a clear need for a methodology supporting the migration of enterprise applications to the cloud. In this work, we focus on enabling support of migration of the database layer of enterprise applications to the cloud. This involves not only considering the requirements on the appropriate data source or service imposed by the application, but also the potential need to adapt the application in order to cope with incompatibilities resulting from the migration. Towards this goal, in this work we presented a step-by-step methodology for application migration. For this purpose, we identified a series of functional and non-functional requirements from the enterprise and eScience domains. We then adapted the methodology discussed in Laszewski and Nauduri (2011) in order to satisfy the identified requirements, which resulted in a seven-step end-to-end methodology for the migration of the database layer of an application to the cloud and for the application refactoring required as part of this process.

Table 4 Documentation of an identified problem

ID	MD 4
Name	Support of special characters in password
Class	Tool (functional correctness, appropriateness)
Severity	Critical
Description	The password for the source or target cloud data store or service does not allow usage of special characters like backslash for instance.
Error handling	Limit the allowed set of characters for the password to digits and alphabetic characters.
Solution/adaptation	As security and privacy is most important today especially in the area of cloud computing the tool has to be extended to support special characters in passwords.

We presented the realisation of the proposed methodology 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 create migration projects, define their requirements for the migrated database layer, describe their current database layer and receive recommendations, hints, and guidelines on where and how to migrate their data. The tool supports conflict resolution based on previously identified cloud data patterns, and provides data adapters that allow for the automatic migration of data to recommended data stores and services. We evaluated the applicability of our approach by migrating the NovaERM Enterprise Resource Management application to an internal Private Cloud of NovaTec and to Amazon Web Services solutions. Apart from the usefulness of the methodology and tool support, we were able to identify missing functionality that we plan to address in our future work. In particular, our evaluation shows that explicit support for the testing phase of the migration has to be supported by the cloud data migration tool. Moreover, the tool should provide sandboxing capabilities, functional testing for bug fixing, and performance benchmarking tools for different application workloads. These capabilities can also be used to support the optimisation of the database layer after its migration. Additional functionalities that are currently being developed 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 and functional suitability of the tool for domain experts. Another important direction for our future research is completing the methodology to enable support for the migration of the other two logical layers of the application architecture.

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) projects 4CaaSt (Grant Agreement No. 258862) and ALLOW Ensembles (Grant Agreement No. 600792) and the German government funded BMBF project ECHO (01XZ13023G).

References

Al-Qutaish, R.E. (2010) 'Quality models in software engineering literature: an analytical and comparative study', *Journal of American Science*, Vol. 6, No. 3, pp.166–175.

Andrikopoulos, V., Binz, T., Leymann, F. and Strauch, S. (2013) 'How to adapt applications for the cloud environment', in *Computing*, Vol. 95, No. 6, pp.493–535, Springer.

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.

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

Cate, F. (1994) 'The EU data protection directive, information privacy, and the public interest', *Iowa L. Rev.*, Vol. 80, p.431, Paper No. 646.

CMMI Product Team (2010) *CMMI for Development*, Version 1.3 (CMU/SEI-2010-TR-033), Software Engineering Institute, Carnegie Mellon University.

Codd, E.F. (1970) 'A relational model of data for large shared data banks', *Communications of the ACM*, Vol. 13, No. 6, pp.377–387.

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

Google, Inc. (2013a) *Google App Engine – Migrating to the High Replication Datastore* [online] http://developers.google.com/appengine/docs/adminconsole/migration (accessed 4 February 2014).

Google, Inc. (2013b) *Google App Engine – Uploading and Downloading Data* [online] http://developers.google.com/appengine/docs/python/tools/uploadingdata?hl=en (accessed 4 February 2014).

Google, Inc. (2013c) *Google Cloud SQL – Importing and Exporting Data* [online] http://developers.google.com/cloud-sql/docs/import_export (accessed 4 February 2014).

ISO/IEC (2005) 'Systems and software engineering – systems and software product quality requirements and evaluation (SQuaRE) – system and software quality models'.

Jamshidi, P., Ahmad, A. and Pahl, C. (2013) 'Cloud migration research: a systematic review', *IEEE Transactions on Cloud Computing*, to appear.

Kan, S.H. (2002) *Metrics and Models in Software Quality Engineering*, Addison-Wesley Longman Publishing Co., Inc., Amsterdam, The Netherlands.

Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J. and Teregowda, P. (2011) 'Decision support tools for cloud migration in the enterprise', in *Proceedings of CLOUD'11*, IEEE, pp.541–548.

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

Leymann, F. et al. (2011) 'Moving applications to the cloud: an approach based on application model enrichment', *International Journal of Cooperative Information Systems (IJCIS)*, Vol. 20, No. 3, pp.307–356.

Louridas, P. (2010) 'Up in the air: moving your applications to the cloud', *Software*, Vol. 27, No. 4, pp.6–11, IEEE.

Menychtas, A. et al. (2013) 'ARTIST methodology and framework: a novel approach for the migration of legacy software on the cloud', in *Proceedings of MICAS'13*, IEEE Computer Society Conference Publishing Services.

Menzel, M. and Ranjan, R. (2012) 'CloudGenius: decision support for web server cloud migration', in *Proceedings of WWW '12*, ACM, pp.979–988.

Microsoft (2013a) *Develop and Deploy with Windows Azure SQL Database* [online] http://social.technet.microsoft.com/wiki/contents/articles/994.develop-and-deploy-with-windows-azure-sql-database.aspx (accessed 4 February 2014).

Microsoft (2013b) *Guidelines and Limitations (Windows Azure SQL Database)* [online] http://msdn.microsoft.com/en-us/library/windowsazure/ff394102.aspx (accessed 4 February 2014).

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

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

Reddy, V.G. and Kumar, G.S. (2011) 'Cloud computing with a data migration', *Journal of Current Computer Science and Technology*, Vol. 1, No. 6, pp.245–257.

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

salesforce.com, Inc. (2013) *Salesforce Help – Data Importing Overview* [online] http://help.salesforce.com/HTViewHelpDoc?id=importing.htm&language=en_US (accessed 4 February 2014).

Shull, F., Carver, J. and Travassos, G.H. (2001) 'An empirical methodology for introducing software processes', *SIGSOFT Softw. Eng. Notes*, Vol. 26, No. 5, pp.288–296.

Sommerville, I. (1996) 'Software process models', *ACM Comput. Surv.*, Vol. 28, No. 1, pp.269–271.

Strauch, S., Andrikopoulos, V., Bachmann, T. and Leymann, F. (2013a) 'Migrating application data to the cloud using cloud data patterns', in *Proceedings of CLOSER'13*, SciTePress, pp.36–46.

Strauch, S., Andrikopoulos, V., Bachmann, T., Karastoynova, D., Passow, S. and Vukojevic-Haupt, K. (2013b) 'Decision support for the migration of the application database layer to the cloud', in *Proceedings of CloudCom'13*, IEEE Computer Society Press, pp.639–646.

Strauch, S., Andrikopoulos, V., Breitenbücher, U., Sáez, S.G., Kopp, O. and Leymann, F. (2013c) 'Using patterns to move the application data layer to the cloud', in *Proceedings of PATTERNS'13*, Xpert Publishing Services (XPS), pp.26–33.

Taylor-Powell, E. and Steele, S. (1996) 'Collecting evaluation data: an overview of sources and methods', University of Wisconsin Cooperative Extension Service, Cir. G3658-4.

Tran, V.T.K., Lee, K., Fekete, A., Liu, A. and Keung, J. (2011) 'Size estimation of cloud migration projects with cloud migration point (CMP)', in *Proceedings of ESEM'11*, IEEE, pp.265–274.

Varia, J. (2010) *Migrating Your Existing Applications to the AWS cloud. A Phase-driven Approach to Cloud Migration* [online] https://s3.amazonaws.com/awsmedia/CloudMigration-main.pdf (accessed 4 February 2014).

Notes

1 NovaTec Holding GmbH: http://www.novatec-gmbh.de/en/.

2 Activiti BPM Platform: http://www.activiti.org.

3 GlassFish Application Server: http://glassfish.java.net.

4 PostgreSQL: http://www.postgresql.org.

5 Amazon S3: http://aws.amazon.com/s3/.

6 Windows Azure SQL Migration Wizard: http://sqlazuremw.codeplex.com.

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

8 Bulk Loader: http://bulkloadersample.appspot.com.

9 Apex Data Loader: http://sforce-app-dl.sourceforge.net.

10 Informatica Cloud: http://www.informaticacloud.com.

11 Microsoft SQL Server: http://www.microsoft.com/en-us/sqlserver.

12 Sybase: http://www.sybase.com.

13 IBM DB2: http://www.ibm.com/software/data/db2.

14 IBM Informix: http://www.ibm.com/software/data/informix/.

15 PostgreSQL: http://www.postgresql.org.

16 Oracle MySQL: http://www.mysql.com.

17 Amazon Relational Database Service: http://aws.amazon.com/rds/.

18 10gen MongoDB: http://www.mongodb.org.

19 Amazon EC2: http://aws.amazon.com/ec2/.

20 Google Cloud SQL: http://cloud.google.com/products/cloud-sql/.

21 Amazon SimpleDB: http://aws.amazon.com/simpledb/.

22 Cloud Data Migration Tool: http://www.cloud-data-migration.com.

23 DataNucleus: http://www.datanucleus.org.

24 automaIT: http://www.automait.de/1/home/.

25 seleniumHQ: http://www.seleniumhq.org.