How to Adapt Applications for the Cloud Environment

Challenges and Solutions in Migrating Applications to the Cloud

Vasilios Andrikopoulos, Tobias Binz, Frank Leymann, Steve Strauch

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

BibTeX

```
@ARTICLE{andrikopoulos2013adapt,

author = {Andrikopoulos, Vasilios and Binz, Tobias and Leymann, Frank and
Strauch, Steve},

title = {How to Adapt Applications for the Cloud Environment},

journal = {Computing},

publisher = {Springer},

volume = {95},

pages = {493-535},

doi = {10.1007/s00607-012-0248-2},

url = {http://dx.doi.org/10.1007/s00607-012-0248-2},

year = {2013}

}
```

Noname manuscript No.
(will be inserted by the editor)

How to Adapt Applications for the Cloud Environment

Challenges and Solutions in Migrating Applications to the Cloud

Vasilios Andrikopoulos · Tobias Binz · Frank Leymann · Steve Strauch

Received: date / Accepted: date

Abstract The migration of existing applications to the Cloud requires adapting them to a new computing paradigm. Existing works have focused on migrating the whole application stack by means of virtualization and deployment on the Cloud, delegating the required adaptation effort to the level of resource management. With the proliferation of Cloud services allowing for more flexibility and better control over the application migration, the migration of individual application layers, or even individual architectural components to the Cloud, becomes possible. Towards this goal, in this work we focus on the challenges and solutions for each layer when migrating different parts of the application to the Cloud. We categorize different migration types and identify the potential impact and adaptation needs for each of these types on the application layers based on an exhaustive survey of the State of the Art. We also investigate various cross-cutting concerns that need to be considered for the migration of the application, and position them with respect to the identified migration types. Finally, we present some of the open research issues in the field and position our future work targeting these research questions.

Keywords Cloud migration · application adaptation · Cloud-enabled applications · data layer · business layer · migration types

1 Introduction

The general motivation for the adaptation of existing applications is to keep and improve the past investments in software while reacting to changes in the environment. The advent and steadily increasing domination of Cloud computing in the software market means that existing applications need to adapt for this environment. Cloud

V. Andrikopoulos
Universitätsstrasse 38, 70569 Stuttgart, Germany
Tel.: +49-711-685-88475
Fax: +49-711-685-88472
E-mail: vasilios.andrikopoulos@iaas.uni-stuttgart.de

computing has become increasingly popular with the industry due to the clear advantage of reducing capital expenditure and transforming it into operational costs [8]. This advantage manifests as the saving of fixed costs by leasing rather than buying infrastructure using the pay-per-use model offered by many Cloud providers. Many applications of course are not ready to be moved to the Cloud because the environment is not mature enough for this type of applications, e.g. safety-critical software [10]. For others, it may not make sense to be migrated at all, e.g. embedded systems. Some software will be implemented specifically for the Cloud (*Cloud-native applications*), but other systems must be adapted to be suitable for the Cloud, making them *Cloud-enabled*. The latter category is the focus of this work.

The emphasis of the existing work is on migrating the whole application based on virtualization technology[1]. For a number of works like [39,50,55,86], the focus is on Virtual Machines (VMs) as the means for migration. This focus can be attributed to the prevalence of the Infrastructure as a Service (IaaS) service model, as defined e.g. by the National Institute of Standards and Technology (NIST) [54], and the early market position of Amazon Web Services[2] that focused on IaaS. Application adaptation in this context manifests on the level of how to manage a dynamic amount of computational resources in trade-off with the cost of these resources. It can actually be argued that one of the major forces behind the Cloud computing growth the last years is exactly this combination of:

- minimum invasiveness to existing application implementations, i.e. simply offloading the whole application stack on a VM and hosting it remotely, and
- delegation of the adaptation concerns to a different level, that of the application resource management.

Beyond this VM-based migration of applications on the IaaS model however, new Platform and Software as a Service (PaaS and SaaS, respectively) offerings of Cloud providers enable alternative migration options for applications. Looking into how applications are usually built, i.e. using the three layers pattern (Presentation, Business logic and Data [26]), as shown in Fig. 1, it can be seen that it is possible to migrate only one architectural layer to the Cloud, instead of the whole application. The Google App Engine for example can be used for the business layer and Amazon Relational Database Service for the data layer. Furthermore, a set of architectural components from one or more layers can also be moved to the Cloud, and different deployment models (Private, Public, Community and Hybrid Clouds, see [54]) can be used, resulting into a *partial migration* of the application.

A number of questions therefore arise: *what part(s) of the application to migrate? How to adapt the application to operate in this mixed environment? Would, at the end of the day, migrating the whole application be a more efficient (in terms of a cost/benefit analysis) solution?* In order to answer these questions we need to understand both how applications that are traditionally deployed (i.e. without using any Cloud technologies, as shown in Fig. 1) should be adapted for the Cloud, and what the impact of this adaptation is to the way that applications usually operate and are

[1] See for example: `http://www.xen.org`, `http://www.vmware.com/products/` and `http://www.flexiant.com/`

[2] Amazon Web Services: `http://aws.amazon.com/`

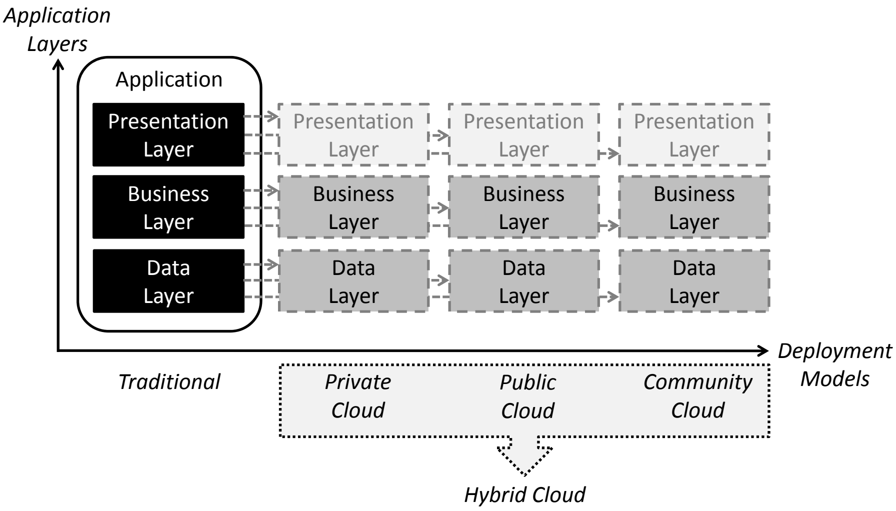

Fig. 1 Overview of Cloud Deployment Models, Application Layers and Possible Migrations

managed. The goal of this work is therefore to identify and discuss research challenges and solutions for adapting existing applications for their migration from the traditional environment of a local data center to the Cloud (i.e. Cloud-enabling them).

The framework of this discussion is structured around the three-layered application architecture pattern as depicted by Fig. 1, and addresses the general principles and challenges that arise on a per-layer and cross-layer basis when migrating the application. In this respect, the challenges arising when considering the migration to a particular service delivery model and/or one specific solution are out of the scope of this work. Furthermore, the focus of the discussion of this article will mostly be the two lower layers of the application architectures (Business and Data). While some of the issues discussed affect also the Presentation layer, we feel that the discussion on how to adapt the Presentation layer of applications has many things in common with transforming applications in services, or at least exposing them as services. This discussion is beyond the scope of this work and has been covered in a better way in the literature, see for example [67].

The contributions of this work can be summarized as follows:

1. an identification and categorization of the various types of application migration to the Cloud,
2. an investigation into how the Business and Data application layer can be migrated to the Cloud, and what type of adaptations are required for this purpose,
3. a survey of cross-cutting concerns that affect both layers, and finally,
4. a systematic positioning of a) the layer-specific challenges and b) the cross-cutting concerns in relation to the identified migration types.

The rest of this article is structured as follows: Section 2 motivates this work by presenting a scenario for which the mere virtualization of the application is not an option. In Section 3 we present our categorization of the various migration types.

Sections 4 and 5 discuss challenges and solutions for migrating architectural components from the Data and Business layer, respectively, to the Cloud. Section 6 presents a series of concerns that affect both layers. Section 7 organizes the findings of the previous sections and positions them with respect to the migration types we identified in Section 2. Finally, Section 8 concludes the article, summarizing the main points and discussing future work.

2 Motivating Scenario

Consider the case of a Health Insurance Company (HIC) in Germany that needs to provide access to its billing data to an External Auditing Company (EAC) for compliance purposes. In particular, HIC must provide EAC with the possibility to query on demand all customer billing transactions in a given time period, without however allowing EAC to access the personal data of the customers. All personal data (e.g. customer name) must therefore be masked before EAC is allowed to query them. Furthermore, HIC must ensure that EAC is not able to reverse-engineer the data masking by e.g. posing more complex queries to their data set.

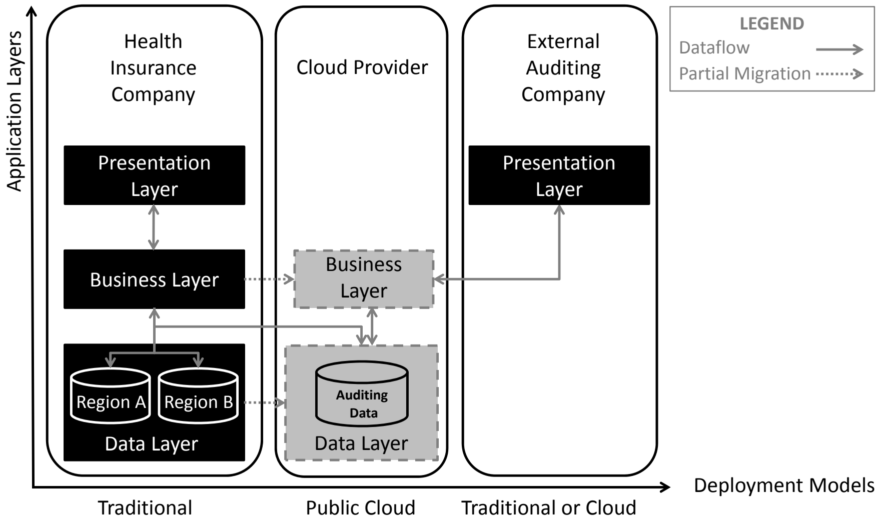

Fig. 2 Example of a Partial Migration of an Application to the Cloud

While HIC is using a distributed topology solution with two data centers for the North and South of Germany (Regions A and B in the bottom left of Fig. 2, respectively), offering direct access to them to EAC creates a series of problems. First, and foremost, the customer and other company-internal data of HIC are the most valuable assets of the company, both for competitive but also for legal accountability reasons. Offering access to the data by means of a standardized interface (operations and queries) that EAC invokes could ensure a degree of security. The performance of the HIC applications running on top of this Data Layer however may be seriously affected by the randomness and load of the query executions by EAC.

Using the oft advertised capability of the Cloud to offer dynamic computational resources on demand would therefore appear to be an obvious choice for an architecture that ensures the required and unhindered operation of both companies. Migrating the whole HIC application stack to the Cloud however is not an option. Beyond the data privacy and legal aspects, HIC has already invested in data centers to store their data. Full migration of the application to a series of VMs, as is the usual practice, is not applicable in this case.

A hybrid solution, like the one described in Fig. 2, is an example solution that would serve the requirements of both HIC and EAC:

- By creating a separate database in the Cloud which only holds the data required for auditing purposes, and moving to the Cloud also the part of the business layer allowing to execute queries on them by EAC, HIC can operate normally.
- By anonymizing the personal data and ensuring synchronization between the "local" and "remote" data, HIC fulfills its obligations w.r.t. EAC and the legal framework governing data privacy for its users.
- The synchronization of data between the two data sets is unidirectional (from the local to the remote) and, as such, its detrimental effect on the performance of HIC's applications is negligible in comparison to the effect of directly executing the queries on the data centers.
- Scaling of the HIC application is decoupled from scaling the part of the application that EAC uses and different strategies can be used if necessary.

3 Migration Types

The scenario described in the previous section illustrates some of the different possibilities for Cloud-enabling an existing application. In order to distinguish between the different approaches, we identify the following *migration types* that Cloud-enable applications through adaptation:

Type I. **Replace** component(s) with Cloud offerings. This is the least invasive type of migration, where one or more (architectural) components are replaced by Cloud services. As a result, data and/or business logic have to be migrated to the Cloud service. A series of configurations, rewiring and adaptation activities to cope with possible incompatibilities may be triggered as part of this migration. Using Google App Engine Datastore in place of a local MySQL database is an example of this migration type.

Type II. **Partially migrate** some of the application functionality to the Cloud. This type entails migrating one or more application layers, or a set of architectural components from one or more layers implementing a particular functionality to the Cloud. Using a combination of Amazon SimpleDB and EC2 instances to host the auditing data and business logic for HIC is an example of such a migration.

Type III. **Migrate the whole software stack** of the application to the Cloud. This is the classic example of migration to the Cloud, where for example the application is encapsulated in VMs and ran on the Cloud. The vast majority of the literature assumes this type of migration as discussed in the introduction.

Type IV. **Cloudify** the application: a complete migration of the application takes place. The application functionality is implemented as a composition of services running on the Cloud. As in the case of component replacement (Type I migration), cloudification requires the migration of data and business logic to the Cloud, in addition to any adaptive actions to address possible incompatibilities.

In the rest of the article we are going to refer to these migration types simply as Type I, Type II, etc. and explicitly name them when necessary. The assumption for each one of these types is that in its initial state, the application is hosted on-premises in a non-Cloud environment, e.g. on a local server, before the identified migration type is applied to it. Migration between Cloud providers and deployment models is therefore beyond the scope of this categorization. Furthermore, while Type III is a monolithic way to Cloud-enable an application by running it as a whole in one or more VMs in the Cloud, Type IV can be considered as a way to make the application Cloud-native. Since however by its definition Type IV does not entail a specific re-engineering for the Cloud environment, the migrated application cannot be truly considered Cloud-native.

4 Data Layer

Security and confidentiality concerns with respect to data migration, e.g. of application data, are one of the main issues impeding the further adoption of Cloud computing in industry and research. In the context of the motivating scenario (Section 2) avoidance of disclosure of HIC-internal business secrets to market competitors by migrating data to the Public Cloud is essential. Hence, usage of Cloud computing in industry is mostly limited to Private Cloud data centers operated by the company utilizing it. For this reason, in the following we identify the research challenges to be addressed by means of investigating the State of the Art of moving the Data Layer to the Cloud. As discussed in the introductory section, not all applications will be moved to the Cloud. However, providing support for the migration of the Data Layer and the necessary adaptations to the application architecture will increase the number of applications that might be moved to the Cloud in the future.

Migration of data can be either seen as the migration of only the Data Layer, or as part of the migration of the whole application. The Data Layer is responsible for data storage and is in turn subdivided into the Data Access Layer (DAL) and Database Layer (DBL). The DAL is an abstraction layer encapsulating the data access functionality. The DBL is responsible for data persistence and data manipulation. The subdivision of the Data Layer leads to a four layer application architecture (Fig. 3).

The migration of the Data Layer to the Cloud includes two main steps to be distinguished for all types of migration: the migration of the DBL to the Cloud, and

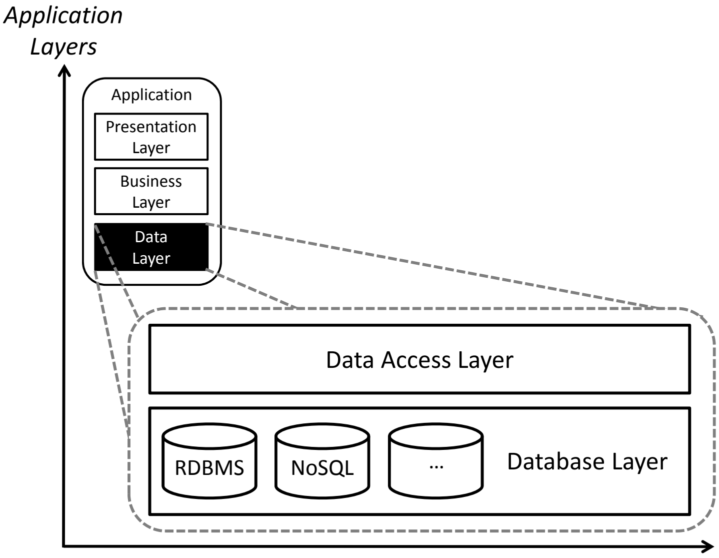

Fig. 3 Subdivision of the Data Layer into Data Access Layer and Database Layer

the adaptation of the DAL to enable Cloud data access. The option to migrate the DBL and adapt the DAL is based on the fact that we consider existing applications for migration that could potentially keep their business logic (partially) on-premises and use more than one Cloud data store providers at the same time. HIC for example is already running a solution with two data centers and a Public Cloud data store is added in order to host the billing data relevant for the compliance audit by EAC. Any statement in this section on the migration is based on the fundamental assumption that the decision to migrate the Data Layer to the Cloud has already been taken. Impact factors and issues to be considered before taking the decision for or against a migration of the whole application or parts of an application to the Cloud such as costs are investigated in Section 6.

We identify the following research questions aiming at addressing the migration of the Data Layer of existing applications:

1. *What* are the possibilities and characteristics of data hosting in the Cloud?
2. *What* are the challenges to the other application architecture layers by distributing the DBL in the Cloud?
3. *What* are reusable solutions for this purpose?
4. *How* to provide transparent data access to the Cloud Database Layer?
5. *How* to provide support for the migration of the Database Layer to the Cloud?

The following sections discuss in detail the challenges in addressing these questions, in combination with presenting related works and identifying open issues.

4.1 Data Hosting in the Cloud

As we assume in this section that the decision to migrate the Data Layer to the Cloud has already been taken, e.g. based on the results of an analysis of cross-cutting concerns such as costs (Section 6), the first step of the migration of the Data Layer is the decision for a specific data hosting solution in the Cloud. Therefore, there is a need for the classification of Cloud data hosting solutions. This will enable a direct comparison of available solutions with respect to Cloud-specific, functional, and non-functional requirements, and provide support when deciding for a specific solution as the first step of migration.

A taxonomy of Cloud computing vendors enabling the comparison between different Cloud services is provided by OpenCrowd [66]. Functional and non-functional aspects are not considered, as the taxonomy focuses on Cloud-specific characteristics and the applications domain of the services. Höfer and Karagiannis [31] introduce a unified business service and Cloud ontology with querying capabilities enabling the mapping of company-internal business functions to offered services in the Cloud. Unifying Cloud services and Cloud providers in an integrated solution, it lacks consideration of non-functional requirements. Kossmann and Kraska [44] analyze the offerings of the major PaaS storage providers like Amazon, Google, and Microsoft and examine the common features and differences by classifying them along three dimensions: deployment type, service type, and supported workloads.

In addition, a categorization of solutions for hosting data in the Cloud has to incorporate the NoSQL solutions that came up in recent years [77]. Architectural decisions on the choice among NoSQL and SQL databases are presented by Hoff [87], but without considering Cloud-related factors like service and deployment model. A list and overview of various available NoSQL databases is provided by Edlich [76]. Two main types of Cloud data hosting solutions can be distinguished with respect to the application interaction with the Cloud data store. The first type allows interaction on a fine granular level, e.g., by using SQL after migrating the database hosted traditionally to an Amazon EC2 instance. The second type provides a service interface to interact with the Cloud data store such as provided by Amazon SimpleDB. The data store becomes a data service, which in turn requires interaction on the level of the service interface that is more coarse grained compared to the interaction when using SQL for instance.

As none of these related works provide decision support for a concrete Cloud data hosting solution incorporating both Cloud-specific as well as data store-related functional and non-functional properties, in [81] we proposed a taxonomy including an initial set of properties. The resulting taxonomy of Cloud Data Hosting Solutions is shown in Fig. 4. We are considering the following six distinguishing properties: *Application Layer* (1 option), *Deployment Model* (4 options), *Location* (2 options), *Service Model* (3 options), *Data Store Type* (2 options) and *Compatibility* (2 options) (Fig. 4). Based on this taxonomy, the term *Cloud Data Hosting Solution* denotes the choice of a concrete option with all six properties considered. This taxonomy supports and guides the user when choosing a provider as the first step when migrating the DBL to the Cloud.

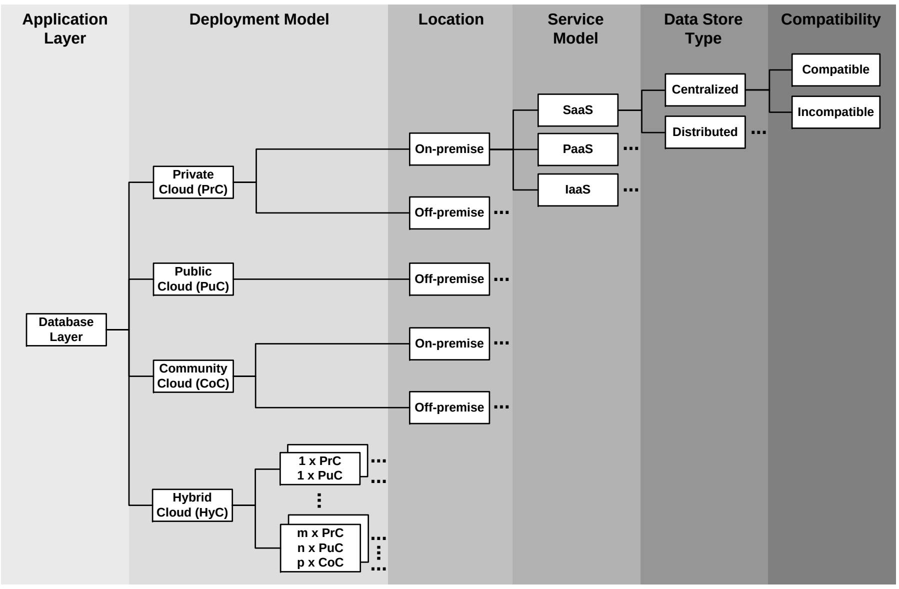

Fig. 4 Taxonomy of Cloud Data Hosting Solutions [81]

4.2 Migrating the Database Layer

Application data is typically moved to the Cloud for the purpose of Cloud bursting, data analysis, or backup and archiving. Hosting the DBL in the Cloud leads to challenges such as *incompatibilities* with the Database Layer previously used, or the accidental disclosing of critical data by, e.g., moving them to a Public Cloud. For example the personal and account information of the customers of the HIC as part of the billing data are considered as critical data in the context of the motivating scenario (Section 2). Thus, its disclosure has to be prevented when migrating the billing data to the Public Cloud. Incompatibilities in the DBL may refer to inconsistencies between the functionality of the traditional DBL used before migration, and the characteristics of an equivalent DBL hosted in the Cloud. For example the Google App Engine Datastore is incompatible with Oracle Corporation MySQL, version 5.5, because the Google Query Language supports only a subset of the functionality offered by SQL, e.g., joins are not supported. Thus, an application making use of such functionality cannot have its Database Layer moved to the Cloud without an impact to its architecture.

In order to provide support when migrating the DBL to the Cloud, and therefore increase the number of applications that might be migrated to the Cloud in the future, we see the need for describing reusable solutions to overcome the recurring challenges such as incompatibilities on an abstract and technology independent level as patterns. Pattern languages to define reusable solutions for recurring problems have been first proposed in architecture by Christopher Alexander et al. [2]. Various publications on patterns exist in Computer Science that provide reusable solutions how to face recurring challenges in different domains.

The challenges and solutions of migrating the DBL to the Cloud and adapting the DAL accordingly have been identified during our work in various EU research projects and especially during the collaboration with industry partners. The identified patterns are also based on literature research focusing on available reports from companies that already migrated their application DBL to the Cloud (Type I migration) and adapted their application accordingly, such as Netflix [6]. Additionally, we take into consideration guidelines and best practices on how to design and build applications in the Cloud, e.g. for enabling scalability [1].

In [78, 80] we propose an initial list of reusable and technology independent solutions for the identified challenges in the form of *Cloud Data Patterns*. A *Cloud Data Pattern* describes a *reusable and implementation technology-independent solution for a challenge related to the Data Layer of an application in the Cloud for a specific context*. So far we have identified three categories of Cloud Data Patterns:

1. *Functional Patterns*
2. *Non-Functional Patterns*
3. *Confidentiality Patterns*

Confidentiality patterns can be considered a subcategory of the non-functional patterns; they are treated separately however due to their importance to data. Table 1 provides an overview of the Cloud Data Patterns we have identified and described so far without claiming completeness of the list of patterns.

In the following we investigate the challenges each of the category of patterns provides solutions for. With respect to their functionality, Cloud data stores and Cloud data services can be considered as appliances that provide a fixed set of functionality [1]. By choosing between SQL and NoSQL for example, each solution is targeting a specific application domain and therefore does not come with all features. The offered functionality might be configurable, but not extensible. *Functional Cloud Data Patterns* provide reusable solutions for these challenges (Table 1). In case the type of data store changes during the migration, e.g., from RDBMS to NoSQL, or BLOB store, it might be not sufficient to emulate or add additional functionality by using Functional Cloud Data Patterns. There may be no equivalent database schema, the consistency model may change from strict to eventual consistency [90], and ACID transactions may not be supported. These are essential conceptual changes and the Business Layer has to be adapted accordingly.

Non-Functional Cloud Data Patterns focus on providing solutions for ensuring an acceptable Quality of Service (QoS) level by means of scalability in case of increasing data read of data write load (Table 1). There are two options for this purpose: vertical and horizontal data scaling [69, 97]. Elasticity with respect to data reads is normally achieved by data replication [19] using read replicas with a master/slave configuration. This is because when write replicas (several master databases) are used, the performance might decrease depending on the consistency model (strict or eventual consistency). In case of strict consistency, a write request to the DBL can only be returned/acknowledged after the data write request has been made persistent to at least $n/2+1$ master database instances, where n is the total number of instances.

Table 1 Overview of Cloud Data Patterns

Category	Name	Icon	Challenge
Functional	**Data Store Functionality Extension**		How can a Cloud data store provide a missing functionality?
	Emulator of Stored Procedures		How can a Cloud data store not supporting stored procedures provide such functionality?
Non-functional	**Local Database Proxy**	P	How can a Cloud data store not supporting horizontal data read scalability provide that functionality?
	Local Sharding-Based Router	R	How can a Cloud data store not supporting horizontal data read and write scalability provide that functionality?
Confidentiality	**Confidentiality Level Data Aggregator**		How can data of different confidentiality levels from different data sources be aggregated to one common confidentiality level?
	Confidentiality Level Data Splitter		How can data of one common confidentiality level be categorized and split into separate data parts belonging to different confidentiality levels?
	Filter of Critical Data		How can data-access rights be kept when moving the Database Layer into the private Cloud and a part of the Business Layer and a part of the data access layer into the public Cloud?
	Pseudonymizer of Critical Data		How can a private Cloud data store ensure passing critical data in pseudonymized form to the public Cloud?
	Anonymizer of Critical Data		How can a private Cloud data store ensure passing critical data only in anonymized form to the public Cloud?

Thus, one write request implies the execution of a number of write requests depending on the number of master databases, which leads to performance degradation.

Confidentiality Cloud Data Patterns provide solutions for avoiding disclosure of confidential data (Table 1). Confidentiality includes security and privacy. With respect to confidentiality, we consider the data to be kept secure and private as critical data such as business secrets of companies, personal data, and health care data, for instance. When migrating pseudonymized or anonymized personal data to the Public Cloud the persons the data is about have to be distinguished from the owners or users of the data. In the motivating scenario (Section 2), HIC migrates billing data relevant for the compliance auditing while filtering its own business secrets to the Public Cloud. EAC is using it for checking the compliance and generating the corresponding auditing reports. Thus, HIC is the owner and EAC is the user of the billing data.

The personal data and account information of the customers as part of the billing data of HIC have to be pseudonymized or anonymized before migrating the billing data to the Public Cloud. As a result, the persons the data is about (HIC customers), and the owner (HIC) and user (EAC) of the data are clearly distinguished. The migration of anonymized or pseudonymized data to the Cloud therefore does not effect the management of the identity of users of Cloud data hosting solutions, e.g. in large-scale Public Cloud environments. The interested reader is referred to [78,80] for an in-depth discussion on these patterns.

Hohpe and Woolf stated that patterns of a pattern language are related to each other, have to be considered as a whole, and must be composable [32]. Thus, we have chosen the form of a piece of a puzzle for the pattern icons in Table 1. As the composability of two or more Cloud Data Patterns depends on their semantics and functionalities, we do not claim that all Cloud Data Patterns are composable with each other. Furthermore, the specific requirements and context of the needed solution may also affect whether a composition of patterns is required.

The usage of Cloud Data Patterns may have a significant impact on the Business Layer. The usage of confidentiality patterns for example can lead to a filtered subset, a pseudonymized, or anonymized form of the data. Moreover, the Emulator of Stored Procedures pattern can also be used to limit the functionality of queries and data manipulation operations allowed by the Business Layer by predefining them as stored procedures. The Business Layer has to be aware of these issues when retrieving data from the Data Layer.

4.3 Cloud-Enabling the Data Access Layer

In this section we discuss the requirements we identified in order to enable the DAL for Cloud data access. In traditional applications built without using any Cloud technology there is in general a tight coupling of the Business Layer with the Database Layer via the Data Access Layer, which implies that the Business Layer is aware of the location of the data and the data store it is interacting with. Especially with respect to Type I and II migrations, where the Database Layer is distributed using non-Cloud technologies and Cloud data stores or data services, we identify the requirement of transparent access of the Business Layer to the data. The distribution of the Database Layer essentially changes the borders of the application (Fig. 5) compared to traditional applications (Fig. 1).

On the one hand, transparent access enables loose coupling between the Business Layer and the Database Layer so that the used Cloud data stores or data services can be changed without affecting the Business Layer. On the other hand, this requires additional functionality in the DAL, because it should be able to determine the data store or data service the request should be forwarded to based on the request sent by the Business Layer.

In case of introducing or using data replication in the DBL in order to increase scalability of reads for instance, the DAL has to be aware of the fact that there are several instances of the DBL, which are synchronized in the background. The required adaptation of the DAL depends on the consistency level to be achieved, i.e. strict

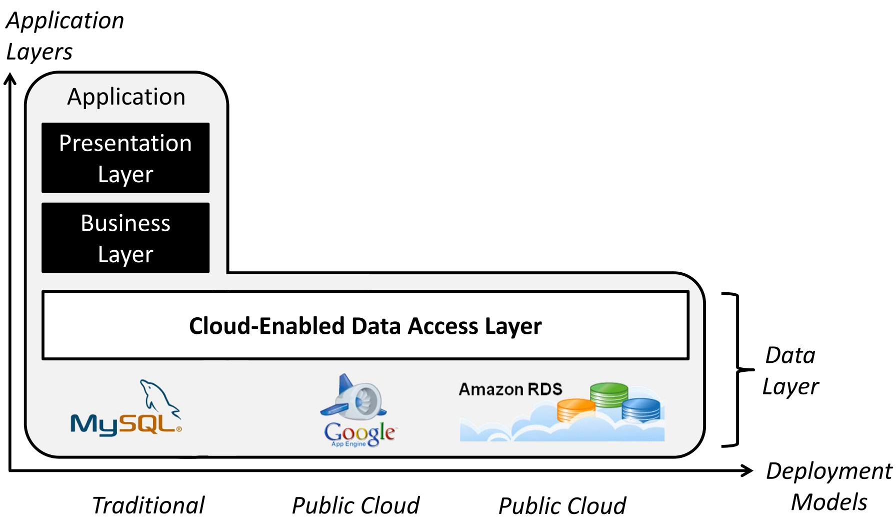

Fig. 5 Change of the Application Border

or eventual consistency [90]. The synchronization mechanism might be triggered by the DAL, for example, in the case of synchronization of an on-premise part of the DBL with the DBL partially moved to the Cloud. Other options in this case include the DBL itself, or external synchronization tools to trigger the synchronization. The billing data of HIC provided to EAC in the Public Cloud part of the DBL has to be consistent with the billing data stored in the two data centers running on-premises.

In addition to incompatibilities with respect to missing functionalities (see functional patterns in Section 4.2), there might be incompatibilities with respect to the semantics of the database schema and/or the database name, e.g., when comparing Oracle with Microsoft SQL Server. These incompatibilities between source and target data store can be overcome by converting between them in the DAL in order to achieve transparency. The same conversion can be applied to challenges with respect to data types that are not supported by the target data store of the migration, e.g., mapping BOOLEAN to BIT or CHAR [47]. Furthermore, in order to enable Cloud data access, the DAL has to enable *reconfiguration* of the data store and/or data service connection. This is sufficient if the source system and the target system of the Database Layer migration are compatible, see Section 4.1.

The requirement for transparent data access of the Business Layer directly implies another important functionality of the DAL. As the Business Layer does not know to which type of Cloud data store or data service the request is forwarded to, the DAL also has to deal with the *transformation of requests*. This affects both the usage of SQL as well as NoSQL data stores. For example, the Google Query Language used to interact with the Google App Engine Datastore supports only a subset of the functionality offered by SQL as joins are not supported. Completely mapping from SQL to NoSQL data stores and transforming the corresponding requests including reconfiguration of the DAL may not always be possible and require some compromises. There exist however experience reports of companies that partially migrated the DBL (Type II) from relational data stores to NoSQL data stores successfully, e.g., from the Web information company Alexa [5] and the media content delivery company Netflix [6].

The difference in the granularity between traditional and Cloud data stores and data services, discussed in Section 4.1, also has an impact to the DAL. The concept of a coarse granular interaction with a Cloud data service did not exist before the advent of Cloud computing and it is fundamentally different, e.g., to the data manipulation and retrieval using SQL. Thus, this imposes a completely new paradigm to be investigated on how to interact with a data store encapsulated behind a service API. The user or client of the data service depends on the functionality and semantics offered by the API, which limits the degree of freedom compared to fine granular interaction via SQL. Hence it has to be investigated what impact this has on the interaction, and how to limit the required adaptations of other applications layers, e.g., by enabling the required functionality in the DAL.

An additional challenge imposed by the migration of the DBL to the Cloud is the potentially high distance between different application layers measured in network hops. The DAL in this environment may have to deal with such issues as network failures in case, e.g., requests to the Cloud data store get lost, and increased latency when accessing or manipulating the data. The impact of the network performance to migrated applications is discussed further in Section 6.5.

Currently, there are several libraries and APIs available that are abstracting from the heterogeneous Cloud provider interfaces, e.g., for managing Cloud instances[3] or using Cloud-specific features during development in Java[4] or Python[5]. These solutions however do not address the requirements identified with respect to Database Layer migration to the Cloud.

4.4 Open Issues

In this section we investigate open issues and other research challenges not being addressed in the previous sections. More specifically:

The Confidentiality Cloud Data Patterns (Section 4.2) enable the filtering, pseudonymization, and anonymization of data when moving it to the Cloud. When considering the migration of the Business Layer to the Cloud, the data will be processed in the Cloud and as such confidentiality patterns are not sufficient to avoid data breaches, e.g., due to attacks from inside the Cloud environment. Those attacks are possible even when using established Cloud providers like Amazon as demonstrated by Ristenpart et al. [71]. A solution for avoiding disclosure of data during the transfer to the Cloud is encryption. The number and type of operations enabled on encrypted data depends on the encryption technique and is limited. Curino et al. propose an onion skin approach by applying different encryption techniques one after another in a sequence before moving the data to the Cloud [23]. When data need to be processed on the Cloud, the encryption layers will be decrypted one by one until the onion skin

[3] Apache Deltacloud: `http://deltacloud.apache.org`

[4] jclouds: `http://www.jclouds.org`

[5] Apache Libcloud: `http://libcloud.apache.org`

with the encryption technique is reached that allows the operation on the encrypted data that are required.

As none of the currently available encryption techniques allow any kinds of operations on encrypted data, fully homomorphic encryption may help to overcome these limitations in the future as shown by Gentry [27]. Fully homomorphic encryption is not yet applicable to real world problems however due to the complexity of determining the results of an encrypted Google request, for instance. Nowadays, providers like Amazon allow to store data in their services in encrypted form. The encryption however does not take place on the client/customer side before moving the data to the Cloud; the service provider is encrypting the data, and can therefore decrypt them whenever needed. Further open issues and security challenges considering all application architecture layers are discussed in Section 6.6.

Another important aspect with respect to trust on the Cloud provider and data confidentiality in the future is how long the data is available in the Cloud and in particular how to reliably erase it. Therefore, the provider has to establish appropriate mechanisms for multi-tenant data management (see Section 6.3), and has to ensure that all replicas, backups, and archives are reliably erased as well. This is of particular interest when a customer has to change the provider, e.g., in case the provider has been acquired by a competitor of the customer. Data deletion is also important in the case of Cloud burst scenarios when the DBL is only temporarily migrated to the Cloud to cover peak loads. The current State of the Art offers data import and data export mechanisms for Cloud services. For example, an engineering team from Google provides support in order to facilitate data import and export to and from Google services[6], but does not provide information on data deletion.

Apart from functional and non-functional aspects to be taken into consideration when migrating the DBL, there are also jurisdictional issues such as compliance. The laws and regulations to be applied depend on the residence of the Cloud service provider company. For example, the US government can enforce the disclosure of customer data from providers resided in the US such as Amazon without notifying the customer in case the national security might be at risk due to the PATRIOT Act [88]. It is irrelevant where the data of the customer is stored, e.g., in Asian or European regions in case of Amazon, as long as the provider company is resided in the US. Additionally, the party responsible for checking for compliance and ensuring that appropriate mechanisms are in place is different depending on the country. In the US, the Cloud service providers are responsible to ensure compliance to law and regulations [51]. In contrast in the EU the Cloud customer is ultimately responsible for investigating whether the provider implements the Data Protection Directive [21].

5 Business Layer

Service-Oriented Architecture (SOA) solutions are widely used in enterprises to overcome integration complexity and reduce management cost [67]. Business functionalities are offered as modular, reusable, self-contained Web services. This includes

[6] The Data Liberation Front: http://www.dataliberation.org

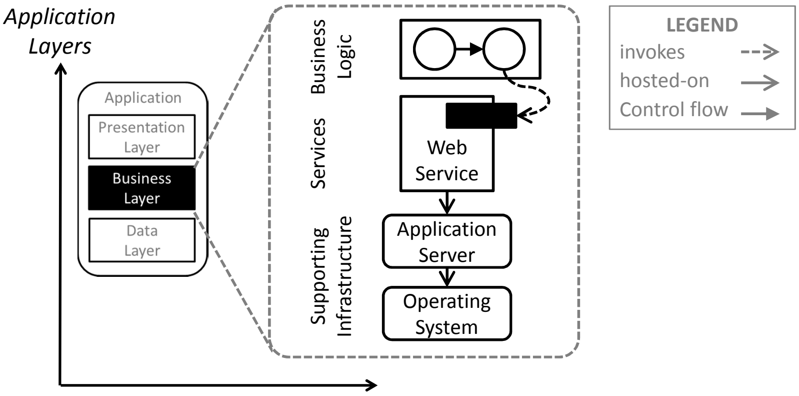

Fig. 6 Example decomposition of BL into business processes, services & their supporting infrastructure

newly developed services as well as existing legacy and external applications, which are wrapped and offered as services. Services are hosted on a *supporting infrastructure* which includes the respective physical resources and software, as shown at the bottom of Fig. 6.

Business processes technology is used to orchestrate multiple services flexibly into higher level business logic, using composition languages like the Business Process Execution Language (BPEL) [62] and Business Process Model and Notation (BPMN) [64], as shown at the top of Fig. 6. Furthermore, business processes may form choreographies denoting the interaction of multiple business partners into an interleaved, coordinated interaction [24]; however choreographies remain currently mainly a research topic. To stay competitive in today's fast changing markets rapid adaptation based on changing business requirements and constant optimization of the business logic is required. One way of doing this is moving business processes and the related services into the Cloud. In the motivating scenario (Section 2) the part of the HIC-internal business processes realizing the business logic to execute queries on the billing data required for compliance checking is partially migrated to the Public Cloud and thus provided to the EAC. Cloud computing enables enterprises to move business processes or parts of them into the Cloud and radically changes the way how services are built, provided, and consumed. Business processes have been mostly deployed in-house, whereas services have also been consumed from third parties. Cloud computing enables new opportunities to migrate processes, services, and their supporting infrastructure into the Cloud.

There already exists a wide range of work on how to adapt business processes [37]. When addressing however infrastructure and management cost, performance, security, greenness, and so on, the properties of the services and its supporting infrastructure directly determine the resulting properties of the business logic [22]. This is due to the fact that a business process is comprised of both the business logic and a number of lower-level services. The majority of the operational and management effort is invested into services and their supporting infrastructure and not on the technical aspects of the business process. For example, Wetzstein et al. [94] show which lower-level service and infrastructure characteristics influence the overall KPIs of the business process. In order to realize adaptation it is key to take services and their supporting infrastructure into consideration [56].

We therefore consider the required adaptation of business logic, services, and their supporting infrastructure to enable the different types of Cloud migration identified in Section 3. Figure 6 shows the three layers we are addressing and their relation. In this section, we address the adaptation of the respective models, i.e., business process models and application topologies, and not the live migration of running instances and their data.

While reflecting on the research in the area of SOA, BPM, and Cloud computing we identified the following research challenges:

1. *How* to create and maintain a fine-grained holistic instance model of an enterprise's IT, which considers, besides the process, also the services and their realization?
2. *How* to ensure manageability of this model and provide the abstraction and extraction methods required for migration of the services' realizations?
3. *How* to analyze and adapt the business process when migrating parts or the whole process to the Cloud?
4. *How* to enable the migration of the service realizations and adapt these applications to use Cloud services?
5. *How* to support a migration model which takes into consideration all layers and components of composite enterprise applications?

In the following sections we investigate each of these research areas, discuss the respective State of the Art, and derive open research issues.

5.1 Integrated Model of Enterprise IT

Workflows and services on the one hand, and their supporting infrastructure on the other, have been investigated extensively but mostly independently, focusing on particular aspects and not in an integrated manner. In particular, services are regarded as black-boxes from the business process point of view and services are implemented by complex composite applications. Therefore, the first group of challenges we identified for Cloud migration is related to an integrated migration model considering all aspects of the BL.

Such a migration model should contain the entire IT infrastructure, software, services, business processes, and their interrelations in one graph. In particular, the parts of IT that are hosted off-premises, for example in the Cloud, should be included to have an overarching model for analysis, optimization, and planning of changes. In the future, this model may also be used to understand the technical and functional relations to business partners, for example, the Cloud provider. Analysing these dependencies is future work not directly related to migration of applications. In general, this decomposition of IT components in a holistic model promotes the bottom up way of thinking. Migration to the Cloud, for example, requires to start from the "bottom", because this is where we find mostly generic and standardized services, which are good candidates for outsourcing [14]. This decomposition also supports the migration of the data layer, as described in Section 4. Business goals, on the other hand, require a top down view of orchestrating lower-level into higher-level services when considered as black boxes. A methodology connecting these two world views is therefore a precondition for adaptation in enterprise IT. On an organizational level, this requires and enables the different groups responsible for the respective levels of enterprise IT and business process to work together more efficiently and be able to facilitate adaptations faster.

Enterprise Architecture Management (EAM) defines the guidelines, design principles, and evolution paths for enterprise IT [18,96] to deal with the increasing complexity problems in enterprise IT today [34]. EAM distinguishes in general five fundamental layers of enterprise IT, namely business, process, integration, software, and infrastructure [96]. A holistic model or view, as proposed here, with respect to all enterprise architecture layers is a prerequisite for the alignment of business goals and IT-related layers EAM is aiming for [18]. However, Winter et al. [95] concludes that no generally accepted model to denote enterprise architectures has evolved yet. In addition, the current state of the enterprise architecture is mostly modeled manually [95], which results in limited information depth and stays on a high granularity level. In [13] we argue that a high-level enterprise architecture should be derived through abstraction from the detailed holistic model of the enterprise IT we are discussing in this section.

Fine grained and technically detailed models to describe composite applications are capable to be automatically deployed to the respective infrastructure. Cafe [57], for example, deploys composite applications modeled using an Eclipse-based modeling tool. The model is based on a graph of nodes representing components and edges of type hosted-on and depends-on. The upcoming OASIS standard TOSCA (*Topology and Orchestration Specification for Cloud Applications*) [63] aims for a portable exchange format describing an application topology, a graph of typed nodes and typed relationships, and the automated management of this application [11]. Machiraju et al. [52] use UML to model application templates and instances. The presented related work captures models representing a single composite application, i.e., classes in the world of programming, for one or multiple interconnected application instances, i.e., objects in object-oriented programming. Enterprise Topology Graphs (ETG) [12] aim to close this gap by capturing a snapshot of the whole enterprise topology as a graph containing all the processes, services, software, infrastructure and their relations. Using ETGs as the integrated model of enterprise IT results in two challenges.

First, using ETGs containing possibly millions of nodes and edges, requires new ways to structure, abstract, and handle this complexity. Depending on the addressed problem domain, tailored views must be provided for the different stakeholders. To migrate business processes and their services to the Cloud it is a prerequisite to identify the respective components for migration and their relations, for example components depending on them. This is facilitated by applying graph processing research which has a long history of providing proven and efficient solutions for graph abstraction. The specific challenges of working on large ETGs are addressed in [13].

The second challenge is the automated creation and updating of an ETG. This can be done by manual modeling, automatic discovery, or importing of existing application descriptions into the ETG. Manual modeling is time consuming, error prone, hard to keep up to date, and, therefore, will lead to a rather informal and coarse-grained model. Such models are not suitable for migration which demands a high degree of technical details and must reflect the current state of the enterprise IT. Importing existing application descriptions is a viable solution, realized through model transformations. However, importing requires that machine-readable application models are available and the model must be complemented with instance information not usually included in the model, for example, the actual IPs and credentials.

Automatic discovery is therefore the most promising approach to create ETGs in practice. In the current State of the Art different approaches are able to discover information from a particular source. For example, *scrawler*[7] identifies dependencies in *Oracle SOA Suite* and NetFlow analyzes network traffic to derive relations between components [20]; similar approaches exist also for specific domains, for example, Java EE applications [36] and storage [35]. Machiraju et al. [52] present a generic approach for application discovery which requires the availability of application template models with the high-level structure of the application to be discovered. These application template models are then refined by agents on the machines.

The discussed approaches do not cover updates of the discovered model. Due to the fact that discarding the model and starting over again regularly is quite expensive, new mechanisms to detect changes and identify the affected areas for re-discovery are research challenges. The challenges for discovery in particular are to enable a discovery which (i) does not depend on application templates existing only for few applications, (ii) is extensible to discover all kind of components, in particular not limited to a certain vendor, (iii) can work without having agents for discovery on the nodes because enterprises are not comfortable with installing additional agents in their production environments, and (iv) able to handle regular changes in the enterprise IT efficiently.

5.2 Adaptation of Business Processes

One motivation to use business processes is their ability to flexibly adapt the business logic. Besides migration to the Cloud, this is required for tasks like outsourcing and insourcing, mergers and acquisitions, and to optimize the business process based on performance indicators (KPIs), for example. These tasks are supported by a variety of tools and operations adapting the business process, like splitting [41] and merging [91], control flow adaptation [74], and adaptation as a reaction to SLA violations [48], as well as data flow analysis [43]. In contrast to adapt and optimize the business process upfront, runtime adaptation of processes is researched, for example, to prevent the violation of service level agreements, as presented by Wetzstein et al. [94].

One challenge is that often no isolated decision can be made on business process level because information of the supporting infrastructure is required, for example, when optimizing for ecological aspects [61]. The same is true for cross-cutting concerns like performance and security as we discuss in detail in Section 6. Approaches

[7] http://code.google.com/p/scrawler/

focusing on the business process but in addition taking the supporting infrastructure into consideration already exist to some extent. Nowak et al. [61], for example, use patterns focusing on ecological aspects to adapt the business process and make predefined changes in the supporting infrastructure required to apply the respective pattern. Based on the existing research, enabling holistic and well-informed decisions based on the ETG is needed.

If, for example, during a Type I or II migration, services have been replaced by or partially migrated to a Cloud hosted service, the business process can act as transparent integration layer for the newly created hybrid application (i.e., the mix of on-premises and Public Cloud business logic in the motivating example). In the literature, Motahari-Nezhad et al. [59], for example, argue to outsource non-core parts of the enterprise IT to the Cloud and use business processes to integrate Cloud and non-Cloud services. This rewiring, however, includes more than changing the endpoints of the business process. For example, enabling the technical connection of functionality hosted in the Cloud with on-premises functionality may require exposing services to the Internet, as well as solving the arising privacy, security, and compliance issues. Besides building upon existing technology, solutions like Amazon Virtual Private Cloud [8], for example, enable customers to securely connect the Amazon Cloud with on-premise IT.

Hosting the whole business process, or parts of it in the Cloud has already been addressed in research. Anstett et al. [7], for example, discuss the possible delivery models of workflow engines in the Cloud and the resulting challenges in the area of security. Optimizing choreographies including on-premise and off-premise business processes is discussed in Wagner et al. [91]. Approaches for distributed execution of business processes, i.e., without requiring a central coordination of business process execution, are presented in [53] based on petri nets and [73] based on peer-to-peer technology. Existing research therefore covers the analysis, optimization, and adaptation of business processes, mostly restricted however to the business process layer. For a more holistic optimization, also taking into consideration the services and supporting infrastructure, the interactions of the layers must also be considered. In addition, the process can serve as means to rewire the adapted application.

5.3 Adaptation of Services and Supporting Infrastructure

Changes done in the business process may require adaptations of the supporting infrastructure, for example, if parts of the business process and the services it orchestrates are to be migrated to the Cloud. The goal is therefore to adapt the application model, while ensuring it has the same or improved functional and non-functional capabilities as before to be able to deploy it into the Cloud. Therefore, the services and supporting infrastructure to migrate must be (i) identified, (ii) extracted, (iii) adapted and optimized, (iv) deployed to the Cloud, and (v) removed from the source environment. The research challenges for each of these steps are discussed in the following.

Based on the Enterprise Topology Graph (ETG) [12] discussed in Section 5.1, and by knowing which services should be migrated based on the adaptations on the

[8] http://aws.amazon.com/vpc/

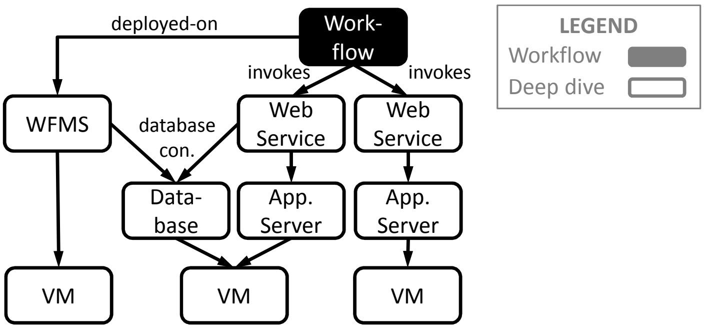

Fig. 7 Visualization of an example workflow deep dive [13]

business layer, the impacted components and required adaptation actions in the composite application must be identified. The *workflow deep dive* technique defined in [13] extracts, for a given workflow represented as node in the ETG, all the services and supporting infrastructure required to fulfill the business process the workflow is implementing. One example of a workflow deep dive is shown in Fig. 7. The open challenge is how to extract the identified parts from the enterprise IT without disrupting other services or applications.

After the components to be migrated have been identified and extracted the actual migration can take place. In [14] we found that many approaches and products for migrations of Type III exist. The *Standardized Format Migration* allows moving, for example, VM images in, e.g., *Open Virtualization Format* (OVF) [85], or applications, e.g., *Java Web Archives* (WAR) [84], which follow a standardized, self-contained format. The *Component Format Migration* transforms one component from one format into another to be able to run on a different environment, for example transforming an OVF to an AMI to run it on Amazon EC2.

Migrations of Type IV, which cloudifies the complete composite application, has only been addressed in research to some extent. For example, CMotion [14] shows how to adapt existing software, based on the application's topology, to use Cloud services instead of on-premise software. Each of the components is migrated while taking into account the relations and dependencies of the components to the others. This may also improve the non-functional requirements of the components, another motivation for the migration of applications into the Cloud.

5.4 Open Issues

After identifying challenges of Cloud migration and reviewing related work we found that a number of research challenges have still not been addressed appropriately. Besides further research in providing a holistic view of the enterprise IT, discovering the enterprise IT model is an open issue, as well as keeping it up to date in an efficient way. For purposes of Cloud migration, business processes cannot be considered isolated from the orchestrated services and their supporting infrastructure. Adapting and optimizing business processes in a holistic way, also taking the services and their supporting infrastructure into consideration, is an open issue. After the parts to be migrated have been identified, the challenge of extracting the components without side effects onto other components arises. Adapting the parts moved to the Cloud to be based on Cloud offerings is another major research challenge. As discussed in Section 5.3, current products and research are focused on the migration of components provided in standardized formats. However, a holistic approach which takes into consideration all layers and components of composite enterprise applications is required.

Increasing IT management and operational complexity, which results in higher IT cost is a challenge for enterprises [34]. Complexity, and with it management cost, increase even more when outsourcing and splitting applications into multiple parts. Due to the fact that we cannot prevent this complexity, new ways to automate and ease management must be found. Therefore, automation of application deployment and in particular management is a challenge to get the growing number of systems under control and enable new ways of adapting IT systems without management costs getting out of control. Automation of management for example is addressed by the upcoming standard TOSCA [11]. In addition, TOSCA addresses portability, a property reducing the complexity of the migration, discussed further in Section 6.7.

6 Cross-Cutting Concerns

The previous sections focused on the Data and Business layers of an application as a way to scope the discussion on how the application may need to be adapted to migrate to the Cloud. In the following section we look at a series of concerns that affect both of these layers, as shown in Fig. 8. The same concerns also affect the Presentation Layer; since this layer is out of the scope of the article however, the discussion will not explicitly acknowledge the impact of these concerns to the application presentation.

Figure 8 summarizes the key concerns that we identify in our discussion, namely:

1. *What* is the impact of the logical and physical distribution of the (migrated) application?
2. *What* elasticity mechanisms can be used for the different types of migration?
3. *How* does migration affect the multi-tenancy capabilities of applications?
4. *How* to calculate the cost of migrating the application and operating in the Cloud?
5. *What* is the impact on the Quality of Service levels of the application, and *how* is application security affected?

While a number of approaches discuss these issues in the literature, as it will be shown in the following, most of the related problems remain open and only partially addressed.

6.1 Logical and Physical Distribution of the Application

Logically and physically distributing the data and computational aspect of applications over the Cloud creates a series of economic, performance, and legal issues for

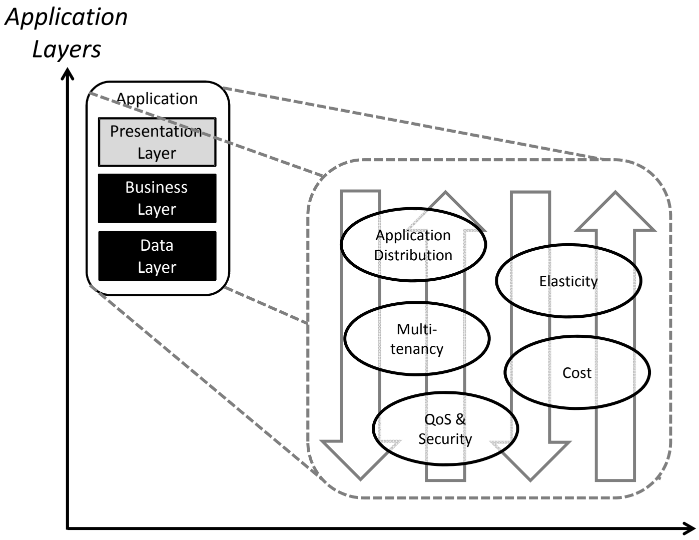

Fig. 8 Application migration cross-cutting concerns

all types of migration. Jim Gray pointed out that $1 buys consistently more computational power than storage space or network bandwidth [28], and concluded that in the general case, it is a good practice to keep data and computation close to each other. Armburst et al.'s update of these calculations showed a clear trend of computational power becoming cheaper faster than disk storage, that becomes cheaper faster than network bandwidth in turn [8]. Following Gray's advice, in [8] it is proposed to physically ship disks with data to the Cloud provider instead of using network uploads. Since this is not always possible, it is critical to identify the needs of the application in terms of computational power and storage space and distribute it accordingly. The *proximity* of the migrated components however, beyond the question of the location of the Cloud provider data centers, may be beyond the control of the application developer as in the case of Type III and IV migrations.

This is because with many Cloud providers it is impossible to constrain the physical or logical location of the application data or logic. As such, it also becomes very difficult to ensure compliance to legal and regulatory requirements concerning for example the privacy of the users. EU regulations for instance require the physical location of the stored data to be inside the EU borders [70], as discussed also in Section 4.4, and applying to the case of HIC in the motivating scenario. Cloud offerings may also be compliant with data protection regulations in specific regions, as in the case of OrionVM for Australia [83], but the same regulations are not applicable outside of their specified region. Even if it is possible to specify coarsely the location of the application, as in the case of Amazon solutions, where it is possible to use their services in one of the 5 defined regions, the cost of using them can be very different due to different charging policies [83]. Furthermore, previous work has shown that Cloud services performance varies significantly for different regions [72], or even for different data centers inside the same region [49]. This issue will be further discussed in Section 6.5.

6.2 Elasticity Mechanisms

Rapid elasticity, the capability to quickly scale outward and inward depending on demand, is one of the essential characteristics of Cloud computing [10]. Ideally, the consumer perceives an infinite number of resources available in any quantity, at any time. Cloud users should be able to avoid excessive costs for over-provisioning applications and loss of revenue in the case of under-provisioning applications, responding to variations in the demand for computational resources. Elasticity provides the means for optimizing resource usage in the case of fluctuating and/or unknown loads. Existing works like [16,83,89], connect the benefits from elasticity mechanisms not only with the Cloud solutions themselves, but also with the particular characteristics of the Cloud-enabled or Cloud-native applications in terms of their work load.

In [89] the authors survey various approaches on application scalability, which is the enabling foundation for elasticity. Two types of scalability are discerned: *horizontal*, adding more instances where required, and *vertical*, adding more computational resources to the application, with the former type being the most common one, at least for IaaS solutions. While vertical scalability is possible in principle for all applications, it largely depends on the service provider to offer the mechanisms to implement scaling dynamically. Horizontal scalability on the other hand mostly depends on the application components and the application as a whole to support it as an option. Applications with demands for high transactionality for example are much more difficult to implement since replicating the data layer between instances requires additional concurrency enforcing mechanisms to be put in place. The billing data hosted within the HIC on-premise data centers for example, have to be made consistent with the billing data migrated to the Public Cloud data store in order to ensure auditing on the latest version of the billing data.

Suleiman et al. [83] identify a series for research challenges affiliated with the elasticity offerings of existing solutions, and in particular:

- *What* to scale, with respect to bottlenecks (called capacity constraints in [70]) like network bandwidth and computing capacity in the performance of the infrastructure.
- *How much* to scale, taking into account automatic scaling mechanisms and the costs associated with licensing (see also Section 6.4 below), and the flexibility enabled by the size and type of resources that can be scaled.
- *How* to scale, and *which* scaling strategy to choose, conducting a trade-off analysis between horizontal and vertical solutions.
- *When* to scale, differentiating between load spikes and long-term work load increase and especially considering *scaling latency* issues exhibited by different providers, as discussed more extensively in [16] and [49].

As discussed in [89], most available IaaS solution providers offer very simple VM management primitives (add/remove VMs), without provisioning for application-related requirements. Scaling is usually triggered by sets of rules/conditions/actions related to VM-specific events or metrics; proposals for application-level scalability are also reported. In addition, the authors of [89] also identify the need for further network-level and PaaS-level scalability research. With respect to the latter in particular, two mechanisms are presented: container-level and database-level scalability. While replication is an obvious candidate for enabling scalability, it may result to degradation effects on the performance of the platform (see also Section 4.3).

Scaling latency, i.e. the time required by the Cloud service to adapt to modified resource needs, is also an important aspect when discussing elasticity. This latency may depend on a number of factors like the service model used (IaaS, PaaS, SaaS), the characteristics and type of the requested resources, the availability of resources and the load of the provider in the region, the rate of load acceleration, and the quotas imposed by the Cloud provider [16]. The authors of [49] for example report faster VM spin-up times for images based on Linux from those based on Windows consistently across different services. Adding VMs to deal with horizontal elasticity is especially susceptible to scaling latency issues.

The NIST report on Cloud computing [10] identifies limitations for the benefits of elasticity depending on different Cloud deployment models. Private Cloud scenarios for example, especially for smaller on-premises deployments, basically exhibit the same limitations in maximum capacity similar to those of traditional data centers. Outsourced Private Clouds are able to provide elasticity only if the Cloud deployment is large enough and there is a sufficient diversity in the applications work load. Public Clouds are generally offering unlimited resources but at a cost. Hybrid architectures, using a combination of traditional and Cloud-enabled computing capabilities, in combination with horizontal scalability are reported in [86] to offer the best solution, at least in terms of cost effectiveness — only however for certain type of applications (in particular, a selection of TPC[9] benchmarks). Further work on this subject is definitely required.

6.3 Multi-tenancy

The multi-tenant model of serving multiple consumers from a common pool of computing resources, including storage, processing, memory and network bandwidth, is one of the essential characteristics of Cloud computing [54]. For the purposes of this discussion we distinguish between two different consumer types with respect to multi-tenancy: *tenants* and *users*. Tenants separate the consumers using a multi-tenant service or application into groups like companies, organizations or departments. These groups are not necessarily completely disjoint since a consumer may belong to more than one tenant at the same time.

Multi-tenancy has been defined in different ways in the literature, see for example [29,45,58,93]. Such definitions however do not address the whole technological stack behind the different Cloud service models as defined in [54] (i.e. IaaS, PaaS and SaaS). For this purpose, in [79] we defined multi-tenancy as *the sharing of the whole technological stack (hardware, operating system, middleware, and application*

[9] The Transaction Processing Performance Council http://www.tpc.org/

instances) at the same time by different tenants and their corresponding users. In this context, *multi-tenant aware* applications and services are the ones that are able to manage and identify multiple tenants and their users, providing tenant-based identification and hierarchical access control to them.

There are two fundamental aspects of multi-tenancy awareness: communication, i.e. supporting message exchanges isolated per tenant, and administration and management, i.e. allowing each tenant to configure and manage individually their communication endpoints at application or service level. *Tenant isolation* is further decomposed into *data* and *performance* isolation between tenants of the same system. Existing approaches on enabling multi-tenancy typically focus on different types of isolation in multi-tenant applications for the SaaS delivery model, see for example [29]. As discussed also in [93] however, only few PaaS (and IaaS) solutions offer multi-tenancy awareness allowing for the development of multi-tenant applications on top of them. The isolation aspect of multi-tenancy, and its implications on the QoS characteristics of a Cloud-enabled application is the subject of ongoing discussion, see for example [3,46]. Furthermore, as it will be discussed further in Section 6.5, the closed nature of the Cloud provides limited visibility to the underlying subsystems and consequently makes the evaluation of isolation difficult.

Discussing multi-tenancy requires that the views of all involved parties are considered, namely both the providers and the consumers of multi-tenant aware services and applications [79]. From the providers' point of view, multi-tenancy allows to maximize the utilization of provided resources and therefore enables maximization of profit. For service consumers, multi-tenancy has to be largely transparent, apart from providing access credentials when using the service or application. More importantly, consumers must have the impression that they are the only ones using the multi-tenant service or application, without suffering from side effects caused by other consumers regarding, e.g., quality of services. Finally, consumers need to be provided with customization capabilities.

Furthermore, the three Cloud service models differ significantly in the granularity of the functionality provided to the consumer, and the required capability of the consumer to manage and control the underlying Cloud infrastructure. The responsibility of the provider and the effort of the consumer to enable multi-tenancy is therefore different, depending on the chosen Cloud service model and the type of the Cloud service. IaaS services offer in principle multi-tenancy only on the resource provisioning level and require from the platforms and/or the applications running in them to implement multi-tenancy on top of them. SaaS services on the other end of the spectrum move the responsibility for multi-tenancy awareness to the service provider and make the underlying resource pooling transparent (in the ideal case) to the application users. A different degree of adaptation may therefore be required to the application depending on the selected type of migration and Cloud service model.

6.4 Cost of Migration and Operation

The cost of migrating an application to a Cloud solution (for any type of migration) and operating at least partially in the Cloud can be decomposed into:

- the *pricing models* of the service providers, and
- the *software licensing & infrastructure procurement costs* imposed by the migration and the elasticity mechanisms discussed in the above.

In the following we discuss how these factors affect the migration and operation cost of Cloud-enabled applications. We also look at existing *cost comparison* tools and methods that can support application owners in estimating these costs and choose the correct Cloud provider.

6.4.1 Pricing models

The discussion in [83] summarizes the various *pricing models* offered by Cloud providers as follows:

- *Per-use*: computing resources in this model are bundled together and billed per unit of time usage. This is the most simple of the models and enables on-demand access to resources at any time without any upfront payments. Prices and price units however can vary between provider offerings and over time periods.
- *Subscription*: computing resources can be reserved in advance by Cloud consumers for a given period of time defined by a signed contract. Some upfront investment is therefore required, which will need to be repeated each time the contract is renewed, often at a discount rate.
- *Prepaid per-use*: the same as the per-use model, but the billing is performed against a pre-paid credit, which if exceeded either the servicing is blocked or charged using the per-use model.
- *Subscription + per-use*: as per the subscription model, dedicated computing resources can be rented for a period of time but additional resources can be requested on demand.

Combinations and variations of these models are also possible: Windows Azure for example offers a 6 month subscription plan that gives a discount on the per-use rate for given usage quotas, beyond which the regular per-use rates apply [10]. Amazon Web Services allow customers to bid for unused EC2 capacity by means of Spot Instances [4]. For typical consumers, the pricing policies are usually non-negotiable and providers reserve the right to change pricing with limited advanced notice [10]. Comparing costs for hosting an application locally or in the Cloud can therefore change significantly over time, as we will see in the following.

6.4.2 Software licensing & infrastructure procurement costs

Software licensing has been identified as one of the major obstacles for migrating to the Cloud [8]. Traditional software licensing is often based on the number of CPUs [70] which does not fit the dynamicity in number of instances and CPUs offered by the Cloud. As such, scaling a system may easily result in unintended license agreement violations. Apart from moving to a licensing model by the CPU-hour, both [8] and [70] recommend using open source software for Cloud purposes, which in many

[10] http://www.windowsazure.com/en-us/pricing/purchase-options/

cases has already provisions for Cloud usage, warning however about having to deal with unsupported versions of such software [70]. Furthermore, in [70] Reese recommends Cloud infrastructure management and procurement operations to be directly connected with the financial department of the consumer in order to make sure that the deployed resources are aligned with the approved budget. This diverges from the traditional model of pushing purchase orders for approval by the financial department when required, and towards an active co-management of the Cloud infrastructure between IT and Finance departments of an enterprise.

In practice, some Cloud providers offer different licensing options to their consumers. For the Amazon Relational Database Service (RDS) for example, consumers can bring their own license for MySQL, Oracle or IBM DB2, get charged for a per-hour license using Oracle DB, or pay a one-time charge per RDS instance to get reduced hourly charging rates [83]. Some companies include licensing fees for free with each account.

6.4.3 Cost comparison

Cost comparison has usually two interrelated aspects: comparing the costs of operating in the Cloud versus operating in a local data center, and comparing the costs of operating in different Cloud providers. With respect to the former aspect, in [8] Armburst et al. pick up the discussion from [28] and enumerate the factors that allow Cloud computing to be more profitable than on-premises computing. In particular, they identify that for cost comparison purposes the option to *pay separately per resource used*, the *power, cooling and building costs of a data center*, as well as the *operations costs* should be considered in the calculation. Only some of these factors are usually taken into account in the literature.

Walker's analysis for example [92], calculates the cost of CPU per hour assuming a Type III migration, taking into account utilization, electricity costs and tax-related depreciation over a period of years. The analysis however focuses on computing power costs without taking into account for example the cost for storage, network and operational costs, and assumes that the work load (expressed as system utilization) is steady over time. The authors of [86] expand and improve this approach to include the possibility to partially migrate only parts of the application to the Cloud (Type II), and incorporate also storage costs in the analysis. More importantly, they propose a classification of the costs associated with migration into two dimensions:

1. *direct* (e.g. hardware and software) and *indirect* (e.g. shared storage, networking infrastructure), and
2. *quantifiable* and *less quantifiable* ones (i.e. costs that are easy or difficult to quantify, respectively).

The analysis presented in [86] focuses on the quantifiable dimension and compares two different Cloud services providers (Amazon and Windows Azure), looking at different application stack deployments using services like Amazon RDS and SQL Azure and different work loads. Based on their experimental results they conclude that full migration is beneficiary only for small or stable organizations; partial migration of an application is too expensive due to high costs of data transfer; and temporary replication of components in the Cloud (Cloud bursting) offers the best value for money for certain applications.

In [70], George Reese looks at what the pay per-use model means for shifting an enterprise's business systems to the Cloud. The discussion is focused around Amazon offerings, and as a consequence the emphasis is on utilizing the IaaS model of service delivery and Type III migration. The key component in his Return of Investment (ROI) and cost comparison analysis is taking into account *hardware depreciation* which in general is two to three years. All costs for using a Cloud infrastructure and the associated management tools, licensing fees, labor costs and third-party setup costs are assessed over a projected three year period and compared against the same costs for setting up, running and maintaining the same infrastructure on premises. Reese's conclusion is that in principle, significant cost savings due to the transfer from capital expenses (CAPEX) to operating expenses (OPEX) incur as the variance increases between peak, average and low capacity of the system.

Various decision support systems for migration of applications to the Cloud [30, 39, 55] have as one of the primary components of the decision making process the cost of operating in the Cloud. They combine the costs of running parts of the whole application in the Cloud (Types I & II) based on the offered pricing model, including networking and storage costs incurred by the application. With respect to comparing the pricing models of various Cloud service providers, Brebner and Liu [17] report on the costs of running applications with different work loads in various Amazon, Google and Microsoft offerings. A similar calculation is performed by [49] and [83] for anonymized Cloud services. The benchmarking of [49] takes into account storage and networking costs, while the analysis of [83] focuses on the elasticity options of the offerings. The actual cost calculation and the conclusions of these comparisons however, while useful, are subject to changes in the pricing and business models of the providers since the time of publication.

Finally, an important issue when estimating the cost of operating on the Cloud is potentially hidden extra charges incurred by Cloud providers. Ingress and egress bandwidth is usually charged separately and at different rates, see for example [25]. These charges are very difficult, or even impossible to predict in advance by the consumer, and extra care has to be taken into investigating their existence before choosing a Cloud provider.

6.5 Quality of Service

QoS dimensions like availability and reliability become very important for the operation of a Cloud-migrated application. The migration of an application to the Cloud entails a loss of control over the QoS characteristics due to the reliance on the QoS levels offered by the service provider. As a result, the QoS characteristics offered by a Cloud service provider appear to have a greater importance to application stakeholders than hosting the application traditionally. There exist two sides in this discussion. On one hand, as discussed in [8], when it comes to availability, few enterprise IT infrastructures can report as good results in terms of outages as the ones by Amazon and Google. On the other hand, while these cases are rare, they may last for hours and have a significant impact on the operation of an application [10]; contingency planning for such cases is recommended to be in place.

Beyond these severe cases of outages, the QoS levels of a migrated application to the Cloud are in principle affected by two major factors: the *performance variability* of the Cloud providers, and the *network latency* between the Cloud service consumers and the service (i.e. the application and the service in the case of Type I and II migrations, and the application consumers and the application itself for Type III and IV migrations). An investigation into the performance variability of major Cloud service providers over a year-long period of time is presented in [33]. High variability is attributed to the combined, non-trivial effects of system size, workload variability, virtualization overheads and resource time-sharing. Two main findings are reported:

1. Performance of Cloud services exhibits yearly and daily patterns of variation, with high variation in monthly median values and periods of stable performance.
2. The impact of performance variability varies significantly across different application types.

Large performance variability for different underlying infrastructure setups even for the same service over the period of one month are also reported by [72]. Dave Durkee explains this variability as a result of the *perfect competition* environment created by service providers translating into practices like resource overcommitment and choosing lower-priced and potentially older infrastructure [25]. Due to this variability, and the lack of visibility to the subsystems constituting a Cloud service, the NIST strongly recommends against using Cloud technologies for safety-critical software [10]. Cloud-migrated application performance is further affected by scaling latency, as discussed also in Section 6.2. The lack of performance isolation due to resource sharing between applications is another factor that contributes to performance variablity. As discussed in [46] however, this is mostly an open issue and suitable metrics for measuring it have to be chosen before appropriate mechanisms are attempted to be identified.

Network latency, both for intra-Cloud networks and Wide Area Networks (WANs), can vary significantly over time and region. [49] for example report very low latency when VM instances are in the same provider data center (as expected). Otherwise, latencies largely correspond to the geographical distribution of the provider data centers. The network topology, provider equipment and location of the data center (with respect to service consumers) also impacts the network latency; good load balancing algorithms may be able to produce near-optimal results in many cases. Traffic shaping and separate charging for fast connections also contribute to bad network latency [25]. In order to overcome potential QoS limitations of WANs like latency up to a certain extend Cloud providers started to offer dedicated network connections to their Cloud network, e.g. Amazon Web Services Direct Connect [11].

In principle, the level of control over the QoS levels of the application decreases for all types of migration due to:

1. application architectures that are not suitable for the Cloud (e.g. low degree of parallelism etc.), and as a result for example they do not scale quickly enough,

[11] http://aws.amazon.com/directconnect/

2. provider performance variability affecting application *performance*, and
3. network performance variability affecting the *latency* of the application.

While application developers can certainly influence the application architecture and the performance aspect (by choosing an appropriate provider), application latency due to the network performance variability is in the general case beyond the control of both the application and the Cloud provider.

Traditionally, QoS levels are guaranteed by service and service level agreements (the latter also known as SLAs). Service agreements are legal documents that specify the rules of the legal contract between providers and consumers, while SLAs describe the technical performace promises made by the provider and actions to be taken in case these promises are broken. As discussed in [10], Cloud providers usually promise to consumers an acceptable availability level in the range between 99.5% an 100%. This range however is calculated on the basis of the billing cycle, and not on the total up-time of the service. Furthermore, failure to provide this availability is compensated with service credit in future use of the services. Additional restrictions may apply on the ratio of this compensation to the actual billed time, and the responsibility to obtain the service credit is with the consumer.

6.6 Security

Last but not least, *security* is one of the major concerns and an obstacle for many enterprises to migrate to the Cloud [8]. For this reason it needs to be addressed separately from the other QoS concerns. Security entails both the *communication* and *data* aspects, but also the *physical/digital* one, i.e., the risk of losing or compromising data due to data center failures or other physical attacks.

Issues like data and network security, intrusion detection and operation in presence of Denial of Service (DoS) attacks are discussed by [40] and [70], focusing on the IaaS delivery model. Subashini and Kavitha [82] provide a survey of the various security risks affiliated with each services delivery model, with an emphasis on the SaaS model. The issues of data security, network security, data segregation in presence of multi-tenancy and management of data breaches are discussed among others. Various security solutions are also presented, and a clear need for a common Cloud security framework is identified and outlined, however it remains future work.

The NIST Cloud Computing Synopsis and Recommendations report [10] also discusses security mechanisms and concerns for each delivery model. In particular about SaaS Clouds, they identify resource sharing as a trade-off between the isolation of the application instances and the efficiency of resource management. The report provides a series of recommendations aimed at Cloud services consumers with respect to security: minimizing consumer (browser) side vulnerabilities, requiring strong encryption and authentication techniques from the providers with visibility into the mechanisms used to enforce them, and considering both physical as well as digital security practices and plans.

As reported in [51], many Cloud providers like Amazon, Google, Microsoft and Force.com have acquired a SAS 70 Type II certification. This means that an independent third party has examined the organizational controls of the provider for processing sensitive information. This however does not ensure in the general case security of communications (usually requiring some kind of secure connection like SSL) and proper data access control mechanisms based on user identification. In this sense, security is a shared concern between Cloud service providers and consumers. On one hand, service providers are required to offer security enabling mechanisms like encrypted communications. On the other hand however, application developers are the ones responsible for adapting the migrated application accordingly to use these mechanisms and further configure it appropriately. Ultimately, it is also the responsibility of the application users to interact with it in a secure manner following security recommendations as the ones discussed in [10]. Overall, security is a serious consideration for both Cloud providers and consumers, and largely an open issue.

6.7 Open Issues

Beyond the issue of security, there are more cross-cutting concerns that are not addressed by the previous discussion because they are mostly open issues both for the academia and the industry. Identifying SLA violations [42] and infrastructures supporting SLA monitoring and enforcement in the Cloud environment [68,83], for example, are beyond the scope of this work. Organizational change as the result or prerequisite of application migration [39] is also not considered.

A number of ongoing works focuses on benchmarking or comparing the performance of Cloud services as the means to allow deciding which Cloud service provider to choose, see for example [16,33,49,72]. Roadmaps for Cloud benchmarking in particular are offered by [3] and [75]. The existing discussion on benchmarking however approaches applications as monolithic entities to be hosted/provided by a Cloud service and focuses on evaluating a particular Cloud service. In this respect the proposed approaches are only suitable for the purposes of Type III migration. Benchmarking applications only partially migrated to the Cloud (Type I or Type II), or composite applications distributed across different Cloud services of the same or different Cloud providers (Type IV), requires a revisit of the existing approaches towards a distributed model of the applications used for benchmarking purposes.

The most important open issue affecting all application layers is probably the interoperability between Cloud service providers. The authors of [60] attribute the difficulties in interoperability between providers to the lack of standardization and the different application models used by services. They come to this conclusion by surveying existing approaches for interoperability between different Cloud service providers, where they also identify a strong emphasis on the IaaS model, at the expense of the other models. Interoperability is also discussed in various degrees in Section 4.4 and Section 5.4, but the discussion is on the level of the needs of each application layer.

A related challenge is portability of the application between Cloud providers. Moving components, or even complete applications, between different providers is often not possible without big investments and rewriting parts of the application. One of the drawbacks of Cloud computing, and external services in general, is lock-in into a specific Cloud management platform, programming framework, or provider.

Besides migration of the components, portability of the deployment and management of the application is another important issue. Today, if the management is automated at all, it is tied to the internals or interfaces of providers or services. This further limits the portability between Cloud providers.

There have been many approaches to describe composite applications [9,57,65] in a programming agnostic-model, but portability of management has not been in the focus until now. Some research has been done in the area of autonomic computing [38] and self-management [15] enabling automation of some management aspects. However, and especially for composite applications, a meta-level management is required which can orchestrate the management capabilities of the components. As an answer to this need, the OASIS standardization initiative TOSCA [63] defines the means for the description of composite applications and portable management [11]. The components of the application topology explicitly define their management operations which are orchestrated into management plans. Based on standards like BPMN and BPEL the management plans are portable to some extent.

7 Discussion

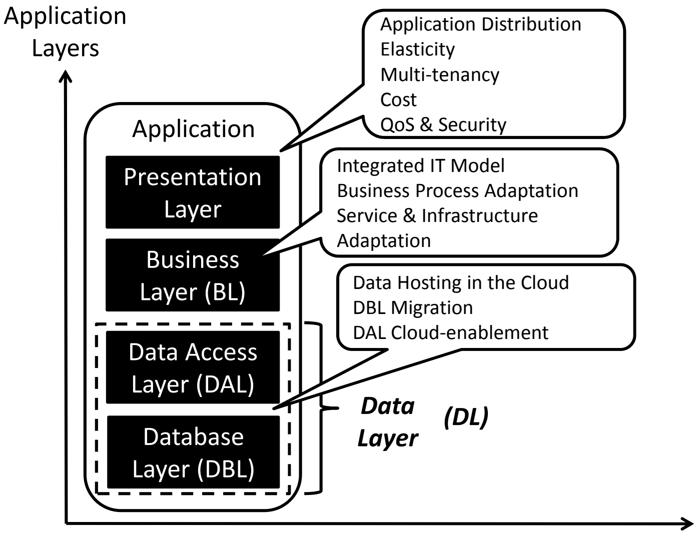

Fig. 9 Summary and positioning of Layer-specific and Cross-cutting concerns

Figure 9 summarizes and positions the challenges raised in the previous sections concerning the Data and Business layers, and across layers. In the following sections we focus on organizing the various mechanisms and issues discussed in the previous sections with respect to the migration types we have identified in Section 2.

Table 2 Layer-specific adaptation actions and migration types

	Adaptation actions	**Type I**	**Type II**	**Type III**	**Type IV**
Business Layer	Adapt BP	–	(√)	–	(√)
	Extract service	–	(√)	–	√
	Adapt service	(√)	(√)	–	(√)
	Redeploy service	–	(√)	–	√
	Rewiring	√	√	–	√
Data Layer	Reconfigure addressing	√	√	–	√
	Realize patterns	(√)	(√)	–	(√)
	Resolve incompatibilities	(√)	√	–	√
	Transform queries	(√)	(√)	–	(√)
	Enable interaction with data store and/or services	√	√	–	√
	Decouple BL from DBL	–	√	–	√
	Legend: √ triggers, (√) may trigger, – no impact				

7.1 Layer-specific Adaptation Actions and Migration Types

Sections 4 and 5 identified throughout their discussion a number of *adaptation actions* that may be required as a result of the application migration. Table 2 summarizes these adaptation actions depending on the migration types that (may) trigger them. In case of Type I and II migrations in particular, the cells in the table refer to the effect of the replacement or partial migration of one or more components in the individual layer only (without their impact across layers). More specifically:

Type I: When an architectural component is replaced by a Cloud service, the application will need rewiring or reconfiguration (in BL and DL, resp.) to use this service. Some adaptation to the new service in the DL may also be required, as is the implementation of a missing functionality, as described by the Cloud Data patterns in Section 4.2. Resolution of incompatibilities and query transformation may take place, depending on the choice of Cloud provider. In any case, the interaction mechanism with the Cloud data store or service will have to be implemented in the DAL of the application by the application developer.

Type II: Partially migrating the BL will definitely need rewiring of the application process and of the underlying services in order to be implemented. Beyond that, the other adaptation actions on the BL level may also be triggered, depending on the functionality which is migrated. For example, if a process is split and a part of it is migrated to the Cloud, then at the very least the business process must be adapted to handle both the migrated and non-migrated parts of the application and rewired appropriately. Similarly, addressing reconfigurations, resolving incompatibilities and enabling the interaction with the Cloud-enabled data store will need to take place. Missing functionalities and query transformation may need to take place, depending on the choice of the provider, as in Type I. Decoupling of the BL from the DBL is especially needed since the DBL is distributed, see Section 4.3.

Type III: As discussed in Section 1, *Type III* migration does not trigger any adaptation actions (under the assumption that the application stack can be extracted and ported as-is to a VM). That is one of the reasons why IaaS solutions are still so popular.

Type IV: Cloudification of the Business Layer requires all architectural components in Fig. 6 to be migrated and provided as Cloud services; as a result, all identified adaptation actions may be required. The business process and the underlying services in particular will need to be adapted if the components in the BL are migrated to different Cloud services. Similarly, all adaptation actions could also be required for the Data Layer. However, depending on the choice of the provider, realizing patterns and transforming queries may be avoided.

7.2 Migration Types and Cross-cutting Concerns

Having established the connection between migration types and adaptation actions on application layer-specific level, in the following section we connect cross-cutting concerns and migration types. By these means we illustrate the potential impact of a migration type to the application beyond layer-specific adaptations. Addressing each one of these concerns may lead to a series of adaptations on one or more of the layers of the application. Identifying which particular adaptations is however beyond the scope of this work.

Table 3 summarizes the concerns raised by Section 6 and positions them with respect to each migration type. More specifically, the *proximity* of the architectural components, affecting both the economics and the performance of the application as discussed in Section 6.1 will decrease due to the network topology (by definition). However, depending on how many providers are selected, and what is the topology of their data centers, the impact will be bigger for a complete cloudification of the application (Type IV) rather than for a simple replacement of one component (Type I). The effect on software *compliance* for all migration types depends completely on the choice of the provider, see Sections 4.4, 5.2 and 6.1.

Based on the dynamic resource allocation capabilities of Cloud providers, the *vertical* scalability of applications overall improves in all cases of migration. The costs however for Type III migration increase significantly in the case of adding bigger

Table 3 Impact of the migration types to the various cross-cutting concerns for the application

		Type I	**Type II**	**Type III**	**Type IV**
Application Distribution	Proximity	–	– / – –	(p)	– –
	Compliance	(p)	(p)	(p)	(p)
Elasticity	Vertical	+	+	+, $r\uparrow$	+
	Horizontal	(c & p)	(c & p)	(a & p), $r\uparrow$	(p)
	Scaling latency	(p)	(p)	(a & p)	(p)
Multi-tenancy	Isolation	(p), –	(p), –	(p), – –	(p), – –
	Admin&Management	(c & p)	(c & p)	(a & p)	(a & p)
Cost	CAPEX/OPEX	(c), –	(c), – –	+	++
	S/W licenses	+	(p & l), –	(p & l), – –	++
QoS	Provider variability	–	–	– –	– –
	Network latency	(n), –	(n), –	(n), – –	(n), – –
Security	Communication	–	–	–	– –
	Data	(c)	(c)	(p)	(p)
	Physical/Digital	(p)	(p)	–	– –

Legend: *p*: depends on provider(s), $r\uparrow$ increases costs, *c*: depends on component(s), *a*: depends on whole application, *l*: depends on license(s), *n*: depends on network, – (–) has a negative impact, + (+) has a positive impact

VMs to accommodate scaling the whole application. Focused scaling on the bottlenecks yields better results in any case. On the other hand, the *horizontal* scalability of the application for the first two migration types depends on the role of the replaced or migrated component for the application (in addition to the capabilities offered by the provider). Migrating performance bottleneck components will produce better results assuming that the provider offers the appropriate scaling mechanisms. If the whole application is migrated, horizontal elasticity is only possible if the application is engineered for these purposes, i.e., it can share its work load between application instances. In any case, the costs will increase as for vertical scalability. *Scaling latency* depends largely on the choice of the provider(s). However, application design is also important, especially for Type III migration.

Given the provider performance variability (Section 6.5) and the security concerns (Section 6.6) of application migration, ensuring performance and data *isolation* becomes a very difficult task for the application developer. Depending on the type of migration, the dependency on the provider to implement appropriate mechanisms to enforce isolation becomes stronger. The overall effect to the isolation capabilities of the migrated application is nevertheless negative. Similarly, the *administration & management* capabilities of the migrated application depend largely on the capacity of the Cloud service provider to support them appropriately. For example, when migrating an ESB to the Cloud in order to offer it as a building block within a Cloud platform [79], this would translate in the capacity of the migrated multi-tenant aware ESB to offer configurable endpoints per tenants. Furthermore, as in the case of elasticity, the application components themselves should be able to support this option.

With respect to cost, the shift from capital to operation expenses (*CAPEX/ OPEX*) is stronger for Types III and IV. For Types I and II however, the cost transfer depends on the role of the architectural components to be migrated. For the latter types, the overall cost may increase significantly due to the costs of operating on two different platforms in parallel (traditionally and on the Cloud). The cost for *software licenses*, as discussed in Section 6.4.2, depends on the provider and the individual licenses of the migrated components. Type III produces the worst results in case no license can be reused. In contrast, Types I and IV incur the least costs since no licenses are required (under the assumption that they are included in the pricing model of the provider).

In terms of the QoS of the application, the *provider performance variability* and *network latency*, as discussed in Section 6.5, will have a definite negative impact to the QoS characteristics of the application. This impact will be bigger for Types III and IV however, due to the complete migration of the application to the Cloud. Finally, security, both for *data* and *communication* and in its *physical/digital* aspect (Section 6.6), degrades in inverse rate to the degree of application migration due to e.g. network vulnerabilities. Depending on the criticality of the migrated architectural component in the case of the Data layer however, the effect may increase significantly.

The following conclusions can therefore be drawn from Table 3:

1. The overall reliance on the provider is clearly higher for Type III; this reliance is reinforced by the vendor lock-in effect.
2. The total cost-benefit ratio is better for Type IV migration. The necessary reengineering effort however, in combination with the number of adaptation actions that may be triggered (Table 2) will deter many application stakeholders from choosing this option.
3. Moving to the Cloud in any form brings security considerations. The bigger the part of the application that is migrated to the Cloud, the higher the security risk, and the higher the demands on the service providers to offer security mechanisms.
4. Any decision making process related to migration must take into account both Cloud service providers, and the role of the to-be migrated architectural component(s) or layers in the overall application architecture.

8 Conclusions & Future Work

Migrating an existing application to the Cloud by encapsulating its software stack in a Virtual Machine (VM) and running it in the Cloud has been very popular with both the industry and academic research. By these means, the adaptation of the application is limited to the way that the application manages its resources. As our analysis shows however, this approach to migration enhances the effects of vendor lock-in, and only works for a limited type of applications.

For this reason, in the previous sections we focused on investigating what types of migrations are available to application developers, and what adaptation actions are required for each of them. With respect to migration types, we categorized them in Types I to IV: Replacement, Partial Migration, Migration of the Whole Software Stack, and Cloudification, respectively. We presented challenges and solutions for the migration of the Business and Data Layers of an application (Fig. 1), on the level of both the whole layer, and that of particular architectural components in each layer. For each migration type we identified its potential impact to the application by correlating them with particular adaptation actions that may be triggered by the migration. We also investigated issues like application distribution, elasticity, multi-tenancy, cost, Quality of Service (QoS) and security that affect both layers and surveyed the State of the Art on them. Furthermore, we positioned each one of these issues in relation to the migration types in order to illustrate their potential impact to the characteristics of the application.

Overall, our analysis shows that migrating to the Cloud, irrespective of the type of migration, will have a negative effect on the security of the application. Beyond that, the cost of operating in the Cloud largely depends on what architectural components of the application are migrated, and it is interconnected with the type of elasticity (vertical or horizontal) to be used for the application. Type IV migration (cloudification of the application) is overall the most effective solution in terms of our analysis. However it requires the most effort for the re-engineering of the application. In many cases of migrating an application the choice of provider is essential.

A very important conclusion from the overall discussion is that there is a clear trade-off between cost and business resiliency. Minimizing the risk (both in terms of security and QoS assurances) of the application migration to the Cloud, requires backup solutions and use of multiple Cloud providers as alternatives (see also [10]). This translates into a multiplication of the costs and requires an application design that takes into account the concerns summarized by Table 3. A more complex analysis than the one presented in Section 7 is required for this purpose, in order to cover combinations of different providers. A further look into the role of the migrated component(s) with respect to function shipping versus data shipping is also necessary. This effort is left as future work.

Furthermore, as discussed in Section 1, the presentation in this work is structured around the three-layered application architecture, with the explicit intention to identify challenges arising in each, and across layers, when migrating the application to the Cloud. When looking however at the migration to a particular delivery model, the identified challenges will naturally materialize as more refined problems. Further challenges may also arise, depending on the particulars of the delivery model or even the specific solution to be used. For the purposes of this work we abstracted away from the particulars of each delivery model and only refer to them when necessary. A reframing of the discussion around the service delivery models is therefore essential for the sake of completeness — as is a repositioning of the potential adaptation effort and impact to cross-cutting concerns with respect to the characteristics and purpose of the application to be migrated.

A holistic methodology of the application migration to the Cloud, which incorporates the various challenges discussed in the previous section and guides application developers through application migration is a natural continuation of this work. This methodology should also provide the means to identify which particular adaptation actions are required given an application architecture, a selection of Cloud providers and a migration type to be applied to it. Furthermore, the analysis and findings provided by the previous sections, especially for Type IV migration, can also be useful for designing Cloud-native applications. Layer-specific issues like provider incompatibilities, and cross-cutting concerns like the effect of elasticity, are essential when discussing applications that are designed to operate in the Cloud. In this case however, further research on how to move between different Cloud deployment models, and a deeper investigation in the interoperability of Cloud providers is also necessary.

Acknowledgements The research leading to these results has received funding from the 4CaaSt project (http://www.4caast.eu) part of the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 258862 and BMWi project CloudCycle (01MD11023). The company, product, and service logos used for identification purposes only. All trademarks and registered trademarks are the property of their respective owners.

The authors would like to thank the reviewers for their insightful comments that contributed towards improving the quality of this work, and Dimka Karastoyanova for her invaluable help and feedback.

References

1. Adler, B.: Building Scalable Applications In the Cloud: Reference Architecture & Best Practices, RightScale Inc. (2011). http://www.rightscale.com/info_center/white-papers/building-scalable-applications-in-the-cloud.php
2. Alexander, C., et al.: A Pattern Language. Towns, Buildings, Construction. Oxford University Press (1977)
3. Alexandrov, A., Folkerts, E., Sachs, K., Iosup, A., Markl, V., Tosun, C.: Benchmarking in the Cloud: What it Should, Can, and Cannot Be. In: 4th TPC Technology Conference on Performance Evaluation & Benchmarking (TPCTC), VLDB 2012 (2012)
4. Amazon Web Services: How AWS Pricing Works (2012). URL http://aws.amazon.com/whitepapers/
5. Amazon.com, Inc.: AWS Case Study: Alexa. URL http://aws.amazon.com/solutions/case-studies/alexa/
6. Anand, S.: Netflix's Transition to High-Availability Storage Systems (2010). URL https://sites.google.com/site/practicalcloudcomputing/index/Netflix%E2%80%99sTransitiontoaKey_v3.1.pdf?attredirects=0&d=1
7. Anstett, T., Leymann, F., Mietzner, R., Strauch, S.: Towards BPEL in the Cloud: Exploiting Different Delivery Models for the Execution of Business Processes. In: Proceedings of the International Workshop on Cloud Services (IWCS 2009) in conjunction with the 7th IEEE International Conference on Web Services (ICWS 2009), pp. 670 –677 (2009). URL http://dx.doi.org/10.1109/SERVICES-I.2009.32
8. Armbrust, M., et al.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009)
9. Arnold, W., Eilam, T., Kalantar, M., Konstantinou, A.V., Totok, A.A.: Pattern Based SOA Deployment. In: Proceedings of the 5th International Conference on Service-Oriented Computing (2007). URL http://dx.doi.org/10.1007/978-3-540-74974-5_1
10. Badger, L., Grance, T., R., P.C., Voas, J.: Cloud Computing Synopsis and Recommendations - Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-146 (2012)
11. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing **16**(03), 80–85 (2012). URL http://doi.ieeecomputersociety.org/10.1109/MIC.2012.43

12. Binz, T., Fehling, C., Leymann, F., Nowak, A., Schumm, D.: Formalizing the Cloud through Enterprise Topology Graphs. In: Proceedings of 2012 IEEE International Conference on Cloud Computing. IEEE Computer Society Conference Publishing Services (2012)
13. Binz, T., Leymann, F., Nowak, A., Schumm, D.: Improving the Manageability of Enterprise Topologies Through Segmentation, Graph Transformation, and Analysis Strategies. In: Proceedings of 2012 Enterprise Distributed Object Computing Conference (EDOC). IEEE Computer Society Conference Publishing Services (2012)
14. Binz, T., Leymann, F., Schumm, D.: CMotion: A Framework for Migration of Applications into and between Clouds. In: Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA). IEEE Computer Society Conference Publishing Services (2011). URL http://dx.doi.org/10.1109/SOCA.2011.6166250
15. Brandic, I.: Towards Self-Manageable Cloud Services. In: Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International, vol. 2, pp. 128 –133 (2009). URL http://dx.doi.org/10.1109/COMPSAC.2009.126
16. Brebner, P.: Is Your Cloud Elastic Enough?: Performance Modelling the Elasticity of Infrastructure as a Service (IaaS) Cloud Applications. In: Third Joint WOSP/SIPEW International Conference on Performance Engineering, ICPE'12, pp. 263–266. ACM (2012)
17. Brebner, P., Liu, A.: Performance and Cost Assessment of Cloud Services. Service-Oriented Computing pp. 39–50 (2011)
18. Buckl, S., Ernst, A., Lankes, J., Matthes, F., Schweda, C.: State of the Art in Enterprise Architecture Management. Tech. rep., Technische Universitt Mnchen, Chair for Informatics 19 (sebis) (2009)
19. Buretta, M.: Data Replication: Tools and Techniques for Managing Distributed Information. John Wiley & Sons, Inc. (1997)
20. Caracas, A., Kind, A., Gantenbein, D., Fussenegger, S., Dechouniotis, D.: Mining Semantic Relations Using NetFlow. In: BDIM'08, pp. 110–111 (2008)
21. Cate, F.: The EU Data Protection Directive, Information Privacy, and the Public Interest. Iowa L. Rev. **80**, 431 (1994)
22. Chandrasekaran, S., Miller, J.A., Silver, G.A., Arpinar, I.B., Sheth, A.P.: Performance Analysis and Simulation of Composite Web Services. Electronic Markets **13**(2) (2003)
23. Curino, C., Jones, E., Popa, R., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N., et al.: Relational Cloud: A Database-as-a-Service for the Cloud (2011)
24. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. Information Technology **50**(2), 122–127 (2008). URL http://dx.doi.org/10.1524/itit.2008.0473
25. Durkee, D.: Why Cloud Computing Will Never Be Free. Queue **8**(4), 20:20–20:29. URL http://doi.acm.org/10.1145/1755884.1772130
26. Fowler, M., et al.: Patterns of Enterprise Application Architecture. Addison-Wesley Professional (2002)
27. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: Proceedings of the 41st annual ACM symposium on Theory of computing, pp. 169–178. ACM (2009)
28. Gray, J.: Distributed Computing Economics. Queue **6**(3), 63–68 (2008). URL http://doi.acm.org/10.1145/1394127.1394131
29. Guo, C., Sun, W., Huang, Y., Wang, Z., Gao, B.: A Framework for Native Multi-Tenancy Application Development and Management. In: Proceedings of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'07). IEEE (2007)
30. Hajjat, M., Sun, X., Sung, Y., Maltz, D., Rao, S., Sripanidkulchai, K., Tawarmalani, M.: Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud. In: ACM SIGCOMM Computer Communication Review, vol. 40, pp. 243–254. ACM (2010)
31. Höfer, C., Karagiannis, G.: Cloud Computing Services: Taxonomy and Comparison. Journal of Internet Services and Applications (2011)
32. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley (2003)
33. Iosup, A., Yigitbasi, N., Epema, D.: On the Performance Variability of Production Cloud Services. In: Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, pp. 104–113. IEEE (2011)
34. Jean-Pierre Garbani Thomas Mendel, E.R.: The Writing on IT's Complexity Wall. Tech. rep., Forrester Research, Inc. (2010)

35. Joukov, N., Pfitzmann, B., Ramasamy, H.V., Devarakonda, M.V.: Application-Storage Discovery. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference, SYSTOR '10, pp. 19:1–19:14. ACM, New York, NY, USA (2010). URL http://doi.acm.org/10.1145/1815695.1815720
36. Joukov, N., Tarasov, V., Ossher, J., Pfitzmann, B., Chicherin, S., Pistoia, M., Tateishi, T.: Static Discovery and Remediation of Code-Embedded Resource Dependencies. In: Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, pp. 233 –240 (2011). URL http://dx.doi.org/10.1109/INM.2011.5990696
37. Karastoyanova, D., Leymann, F.: Making Scientific Applications on the Grid Reliable Through Flexibility Approaches Borrowed from Service Compositions. Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications, Information Science Publishing, United Kingdom (2009)
38. Kephart, J., Chess, D.: The Vision of Autonomic Computing. Computer **36**(1), 41 – 50 (2003). URL http://dx.doi.org/10.1109/MC.2003.1160055
39. Khajeh-Hosseini, A., Greenwood, D., Smith, J.W., Sommerville, I.: The Cloud Adoption Toolkit: Supporting Cloud Adoption Decisions in the Enterprise. Software: Practice and Experience **42**(4), 447–465 (2012)
40. Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., Teregowda, P.: Decision Support Tools for Cloud Migration in the Enterprise. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 541–548. IEEE (2011)
41. Khalaf, R., Leymann, F.: Role-based Decomposition of Business Processes using BPEL. In: International Conference on Web Services (ICWS 2006), pp. 770–780. IEEE Computer Society (2006). URL http://dx.doi.org/10.1109/ICWS.2006.56
42. Klems, M., Nimis, J., Tai, S.: Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. Designing E-Business Systems. Markets, Services, and Networks pp. 110–123 (2009)
43. Kopp, O., Khalaf, R., Leymann, F.: Deriving Explicit Data Links in WS-BPEL Processes. In: Proceedings of the International Conference on Services Computing, SCC 2008, pp. 367–376. IEEE Computer Society Press, Honolulu, Hawaii, USA (2008). URL http://dx.doi.org/10.1109/SCC.2008.122
44. Kossmann, D., Kraska, T.: Data Management in the Cloud: Promises, State-of-the-art, and Open Questions. Datenbank Spektrum **10**(3), 121–129 (2010)
45. Krebs, R., Momm, C., Konev, S.: Architectural Concerns in Multi-Tenant SaaS Applications. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science (CLOSER'12). SciTePress (2012)
46. Krebs, R., Momm, C., Kounev, S.: Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments. In: B. Buhnova, A. Vallecillo (eds.) Proceedings of the 8th International ACM SIGSOFT Conference on the Quality of Software Architectures, CBSE'12, pp. 91–100. ACM Press, New York, USA (2012)
47. Laszewski, T., Nauduri, P.: Migrating to the Cloud: Oracle Client/Server Modernization. Syngress (2011)
48. Leitner, P., Wetzstein, B., Karastoyanova, D., Hummer, W., Dustdar, S., Leymann, F.: Preventing SLA Violations in Service Compositions Using Aspect-based Fragment Substitution. Service-Oriented Computing pp. 365–380 (2010)
49. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: Comparing Public Cloud Providers. In: Proceedings of the 10th Annual Conference on Internet Measurement, IMC '10, pp. 1–14. ACM, New York, NY, USA (2010). URL http://doi.acm.org/10.1145/1879141.1879143
50. Lloyd, W., Pallickara, S., David, O., Lyon, J., Arabi, M., Rojas, K.: Migration of Multi-tier Applications to Infrastructure-as-a-Service Clouds: An Investigation Using Kernel-Based Virtual Machines. In: Proceeding of the 12th IEEE/ACM International Conference on Grid Computing (GRID 2011), pp. 137–144. IEEE (2011)
51. Louridas, P.: Up in the Air: Moving Your Applications to the Cloud. Software, IEEE **27**(4), 6–11 (2010)
52. Machiraju, V., Dekhil, M., Wurster, K., Garg, P., Griss, M., Holland, J.: Towards Generic Application Auto-Discovery. In: IEEE/IFIP Network Operations and Management Symposium (2000). URL http://dx.doi.org/10.1109/NOMS.2000.830376
53. Martin, D., Wutke, D., Leymann, F.: A Novel Approach to Decentralized Workflow Enactment. In: Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC 2008). Munich, Germany, September 15 - 19, 2008., pp. 127–136. IEEE Computer Society (2008)

54. Mell, P., Grance, T.: Cloud Computing Definition. National Institute of Standards and Technology (NIST) (2009)
55. Menzel, M., Ranjan, R.: CloudGenius: Decision Support for Web Server Cloud Migration. In: Proceedings of the 21st International Conference on World Wide Web, WWW '12, pp. 979–988. ACM, New York, NY, USA (2012). URL http://doi.acm.org/10.1145/2187836.2187967
56. Mietzner, R., Fehling, C., Karastoyanova, D., Leymann, F.: Combining horizontal and vertical composition of services. In: Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on, pp. 1–8. IEEE (2011)
57. Mietzner, R., Unger, T., Leymann, F.: Cafe: A Generic Configurable Customizable Composite Cloud Application Framework. In: R. Meersman, T. Dillon, P. Herrero (eds.) CoopIS 2009 (OTM 2009), *Lecture Notes in Computer Science*, vol. 5870, pp. 357–364. Springer-Verlag, Berlin, Heidelberg (2009)
58. Mietzner, R., et al.: Combining Different Multi-Tenancy Patterns in Service-Oriented Applications. EDOC'09. IEEE (2009)
59. Motahari-Nezhad, H.R., Stephenson, B., Singha, S.: Outsourcing Business to Cloud Computing Services: Opportunities and Challenges. IEEE IT Professional, Special Issue on Cloud Computing **11** (2009)
60. Nguyen, D.K., Taher, Y., Papazoglou, M.P., van den Heuvel, W.J.: Service-based Application Development on the Cloud - State of the Art and Shortcomings Analysis. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science (CLOSER'12). SciTePress (2012)
61. Nowak, A., Binz, T., Fehling, C., Kopp, O., Leymann, F., Wagner, S.: Pattern-driven Green Adaptation of Process-based Applications and their Runtime Infrastructure. Computing pp. 1–25 (2012). URL http://dx.doi.org/10.1007/s00607-012-0188-x
62. OASIS: Web Services Business Process Execution Language Version 2.0 – OASIS Standard (2007)
63. OASIS: Topology and Orchestration Specification for Cloud Applications Version 1.0 Working Draft 06 (2012). URL http://www.tosca-open.org
64. Object Management Group (OMG): Business Process Model and Notation (BPMN) Version 2.0 (2011). URL http://www.omg.org/spec/BPMN/2.0/. OMG Document Number: formal/2011-01-03
65. OMG: Unified Modeling Language (UML) (2011). http://www.omg.org/spec/UML
66. OpenCrowd: Cloud Computing Vendors Taxonomy. URL http://cloudtaxonomy.opencrowd.com/
67. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. Computer **40**, 38–45 (2007). URL http://doi.ieeecomputersociety.org/10.1109/MC.2007.400
68. Patel, P., Ranabahu, A., Sheth, A.: Service Level Agreement in Cloud Computing. In: Cloud Workshops at OOPSLA (2009)
69. Pritchett, D.: BASE: An ACID Alternative. Queue **6**(3), 48–55 (2008)
70. Reese, G.: Cloud Application Architectures. O'Reilly Media, Inc. (2009)
71. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS) pp. 199–212 (2009)
72. Schad, J., Dittrich, J., Quiané-Ruiz, J.: Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. Proceedings of the VLDB Endowment **3**(1-2), 460–471 (2010)
73. Schuler, C., Weber, R., Schuldt, H., Schek, H.J.: PeertoPeer Process Execution with Osiris. In: M. Orlowska, S. Weerawarana, M. Papazoglou, J. Yang (eds.) Service-Oriented Computing - ICSOC 2003, *Lecture Notes in Computer Science*, vol. 2910, pp. 483–498. Springer Berlin / Heidelberg (2003)
74. Sonntag, M., Karastoyanova, D.: Ad hoc Iteration and Re-execution of Activities in Workflows. International Journal On Advances in Software **5**(1 and 2), 91–109 (2012)
75. SPEC Open Systems Group - Cloud Computing Working Group: Report on Cloud Computing to the OSG Steering Committee (2012). URL http://www.spec.org/osgcloud/docs/osgcloudwgreport20120410.pdf
76. Stefan Edlich: List of NoSQL Databases (2011). URL http://nosql-database.org
77. Strauch, C.: NoSQL Databases (2011). URL http://www.christof-strauch.de/nosqldbs.pdf. http://www.christof-strauch.de/nosqldbs.pdf
78. Strauch, S., Andrikopoulos, V., Breitenbücher, U., Kopp, O., Leymann, F.: Non-Functional Data Layer Patterns for Cloud Applications. In: Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom'12). IEEE Computer Society Press (2012)

79. Strauch, S., Andrikopoulos, V., Gómez Sáez, S., Leymann, F., Muhler, D.: Enabling Tenant-Aware Administration and Management for JBI Environments. In: Proceedings of the 5th International Conference on Service-Oriented Computing and Applications (SOCA'12). IEEE Computer Society Conference Publishing Services (2012)
80. Strauch, S., Breitenbücher, U., Kopp, O., Leymann, F., Unger, T.: Cloud Data Patterns for Confidentiality. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science (CLOSER'12). SciTePress (2012)
81. Strauch, S., Kopp, O., Leymann, F., Unger, T.: A Taxonomy for Cloud Data Hosting Solutions. In: Proceedings of the International Conference on Cloud and Green Computing (CGC '11). IEEE Computer Society (2011). URL http://dx.doi.org/10.1109/DASC.2011.106
82. Subashini, S., Kavitha, V.: A Survey on Security Issues in Service Delivery Models of Cloud Computing. Journal of Network and Computer Applications **34**(1), 1–11 (2011)
83. Suleiman, B., Sakr, S., Jeffery, R., Liu, A.: On Understanding the Economics and Elasticity Challenges of Deploying Business Applications on Public Cloud Infrastructure. Journal of Internet Services and Applications pp. 1–21 (2011)
84. Sun Microsystems: JSR 154: Java Servlet Specification, Version 2.5 (2007). URL http://jcp.org/en/jsr/detail?id=154
85. System Virtualization, Partitioning and Clustering Working Group: Open Virtualization Format Specification (DSP0243). Distributed Management Task Force (2009)
86. Tak, B., Urgaonkar, B., Sivasubramaniam, A.: To Move or not to Move: The Economics of Cloud Computing. In: Third USENIX Workshop on Hot Topics in Cloud Computing (HOTCLOUD 2011) (2011)
87. Todd Hoff: 35+ Use Cases for Choosing Your Next NoSQL Database (2011). URL http://highscalability.com/blog/2011/6/20/35-use-cases-for-choosing-your-next-nosql-database.html
88. US Congress: Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT Act) Act of 2001. Enrolled Bill [Final as Passed Both House and Senate] - ENR (2001). URL http://thomas.loc.gov/cgi-bin/t2GPO/http://www.gpo.gov/fdsys/pkg/BILLS-107hr3162enr/pdf/BILLS-107hr3162enr.pdf
89. Vaquero, L., Rodero-Merino, L., Buyya, R.: Dynamically Scaling Applications in the Cloud. ACM SIGCOMM Computer Communication Review **41**(1), 45–52 (2011)
90. Vogels, W.: Eventually Consistent. Communications of the ACM **52**(1), 40–44 (2009). URL http://portal.acm.org/citation.cfm?id=1435432
91. Wagner, S., Kopp, O., Leymann, F.: Towards Choreography-based Process Distribution In The Cloud. In: Proceedings of the 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pp. 490–494. IEEE Xplore, Bejing, China (2011). URL http://dx.doi.org/10.1109/CCIS.2011.6045116
92. Walker, E.: The Real Cost of a CPU Hour. Computer **42**(4), 35–41 (2009)
93. Walraven, S., et al.: A Middleware Layer for Flexible and Cost-Efficient Multi-tenant Applications. Middleware'11 (2011)
94. Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., Leymann, F.: Monitoring and Analyzing Influential Factors of Business Process Performance. In: Proceedings of the IEEE International Enterprise Distributed Object Computing Conference (EDOC '09), pp. 141 –150 (2009). URL http://dx.doi.org/10.1109/EDOC.2009.18
95. Winter, K., Buckl, S., Matthes, F., Schweda, C.M.: Investigating the State-of-the-Art in Enterprise Architecture Management Methods in literature and Practice. In: MCIS, p. 90 (2010)
96. Winter, R., Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In: Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops, EDOCW '06, pp. 30–. IEEE Computer Society, Washington, DC, USA (2006). URL http://dx.doi.org/10.1109/EDOCW.2006.33
97. Zawodny, J., Balling, D.: High Performance MySQL: Optimization, Backups, Replication, Load-balancing, and More. O'Reilly & Associates, Inc. Sebastopol, CA, USA (2004)

All links were last followed on November 26, 2012.