The Benefits of Using Experimental Exploration for Cloud Migration Analysis and Planning

Frank Fowley[1], Divyaa Manimaran Elango[1], Hany Magar[1], and Claus Pahl[2(✉)]

[1] IC4, Dublin City University, Dublin 9, Ireland
[2] SwSE, Free University of Bozen-Bolzano, Bolzano, Italy
claus.pahl@unibz.it

Abstract. Migration software systems to the cloud causes challenges. This applies especially for companies that do not have sufficient cloud expertise. In many of these companies there is a clear ideas about expected benefits. There is also an awareness of some potential problems. However, this is often not sufficient to assess the risks before starting on a full cloud migration of a legacy system.

Technical and conceptual analyses can only help to identify risks in the migration process with from a cost and a quality perspective to a limited extent. So, we investigate here the suitability of feasibility studies with a focus on experimental exploration. These studies would generally only cost 5% of the overall costs of a migration project, but can strongly support a reliable risk assessment. These can determine how much of the expectations and intentions can achieved in a cloud deployment. The cost of the migration, but also the cost of operating an IT system in the cloud can be estimated in the context of quality expectations. Using a feasibility study with an experimental core based on a partial prototype delivers much more reliable figures regarding configurations, quality-of-service and costing than a theoretical analysis could deliver.

We will embed our feasibility study approach into a pattern-based migration method. We report on a number of case studies to validate the expected benefits of feasibility-driven migration.

Keywords: Cloud migration · Experiment · Prototyping · Migration patterns
Cloud architecture · Cloud cost model · Performance · Scalability

1 Introduction

Today, cloud computing is a widely used form of operating software. However, the migration of software systems to the cloud [2] is still a problem for many, especially small and medium-sized enterprises and organisations without sufficient cloud expertise [12]. These companies generally need to rely on support from consultants and solution providers. Expected benefits are known and an awareness of potential problems exists. However, this is often not sufficient to confidently embark on a full migration of a software system, which requires estimates of the migration costs as well as costs of operating software within the limits of required quality in the cloud.

D. Ferguson et al. (Eds.): CLOSER 2017, CCIS 864, pp. 157–176, 2018.
https://doi.org/10.1007/978-3-319-94959-8_9

Technical and cost-oriented discussions can help to some extent, but many assumptions rely on expert knowledge from similar cases, but might not always be fully reliable. We investigate here feasibility studies with a focus on experiments to assess risk through quality and cost analyses. These studies cost typically be five percent of the overall migration cost. They can consequently be considered a worthwhile investment.

Apart from legacy-to-cloud migration, our solution can also be used for migrations to another cloud architectures. So, rather than cloud-onboarding, the source architecture is already cloud-based. An IaaS to PaaS migration from a basic, virtualized on-premise system into a fully cloud-native architecture could serve as an example here.

An adequate project scoping for any cloud migration is a problem [4], because of misconceptions and unclear technical expectations that significantly increase the risk. Migration frameworks and case studies have been reported in the literature by academics and practitioners from industry [2,3,18], but how to reliably estimate a 'right-scaling' of a cloud deployment remains unclear if cost and quality concerns need to be aligned.

Feasibility studies can support risk assessment. It can clarify how much of the expectations and intentions can be achieved. The cost of the migration [5] and also the cost of operating a software system in the cloud can be better estimated [6]. It also helps to better understand technical cloud architecture concerns. Another question is a re-engineering one. What is migratable and what is the extent of refactoring necessary to make migration work are questions. Often, re-engineering software in order to modernise and adapt to cloud constraints is needed. Prototyping in the cloud can help to address scalability requirements, i.e., to align performance and cost concerns. Using a feasibility study with an experimental core based on a partial prototype of the proposed cloud software system [13] delivers much more reliable figures regarding configurations, quality-of-service and costing than a theoretical analysis could delivery.

In this investigation, we report on case studies that we carried out as independent consultants with small and medium-sized enterprises in the software sector. These case studies serve to validate the benefits of the approach towards a more reliable risk analysis by integrating cost and quality.

The paper is structured as follows. We introduce a migration feasibility framework in Sect. 2. The, in Sect. 3, we define our architecture assessment and risk framework that links to known architecture analysis methods. We investigate the goals of feasibility studies and practical concerns of feasibility studies in a migration project in Sect. 4. We focus on the experimentation process in Sect. 5. In Sect. 6, we discuss a selected use case in more detail. In Sect. 7, we discuss our observations. Related work is discussed in Sect. 8, before ending with conclusions and a discussion of future work.

2 A Pattern-Based Migration Planning Framework

Our migration approach starts with a pattern-based architecture description. These are used to define an initial migration plan.

2.1 Pattern-Based Cloud Architecture Migration

Our proposed pattern-based architecture migration method considers the following components: (i) the source architecture of the system, (ii) the target architecture options in the cloud, and (iii) the high-level architectural transformations for the different target architectures [7]. This is formalised through a catalogue of migration patterns which each describe simple architectural transformations for specific scenarios (e.g. for simple cloudification in an IaaS solution). This defines a staged process based on a migration path which in the individual steps are driven by selection criteria (e.g., time to market or introduction of new capabilities). A sample pattern is Multi-Cloud Relocation [6], see Fig. 1, which simply replaces on-premise by cloud-based components.

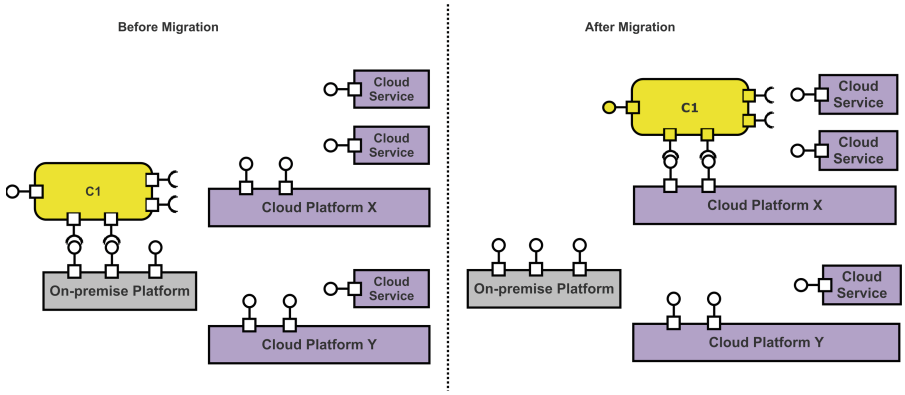

Fig. 1. Cloud migration pattern: multi-cloud relocation.

Several migration patterns are sequenced together to form an incremental migration path, see Fig. 2 which gives a path with options included. Usually, a cloud migration is an incremental activity, which is reflected in the path.

A concrete application shall also be introduced. A sample migration from a legacy system's architecture to a cloud-based one is displayed in Figs. 3 and 4. In this case, the migration of a classical enterprise application, an expense system, is migrated into the cloud, ultimately using several cloud-native services (here Azure storage services), but also other external services (such as the Payment component).

Of key importance in a migration are:

- the expected quality-of-service;
- the resources needed;
- the architecture patterns meant to be preserved or employed;
- the platforms chosen to host the application in the cloud.

These need to be mapped to patterns.

A migration pattern is represented by an architecture diagram of the service architecture deployment before and after migration, i.e., a migration pattern is a transformation triple based on source and target architecture combined with the applied pattern

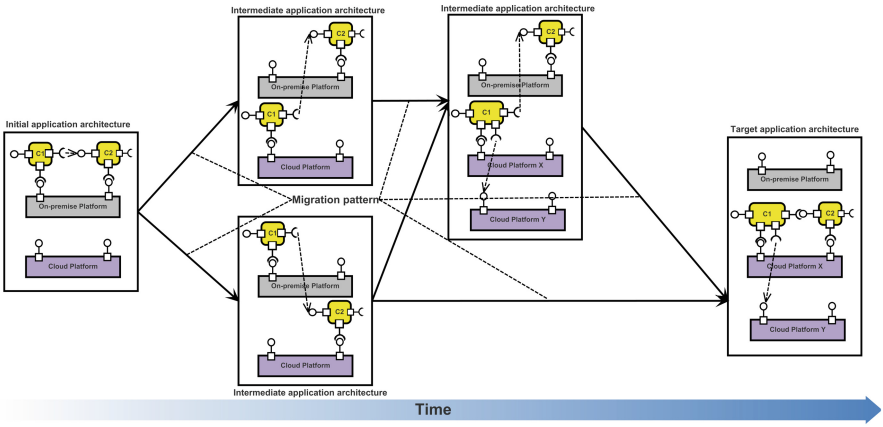

Fig. 2. Migration path based on several pattern applications.

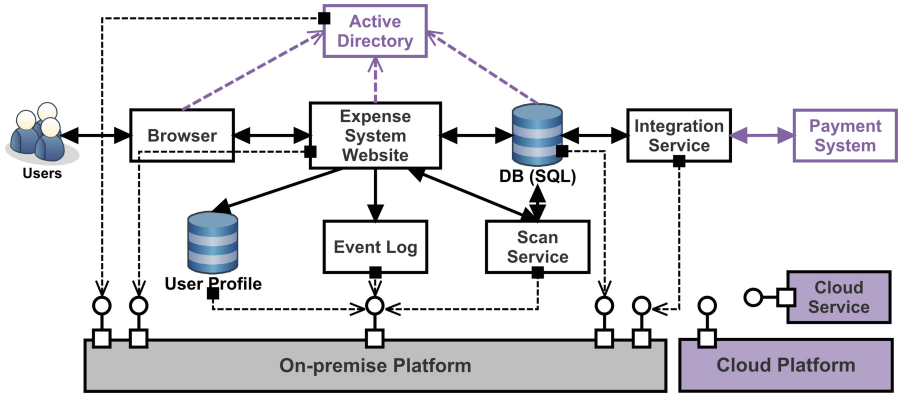

Fig. 3. Sample migrated architecture – before migration.

as the transformation specification. Each source or target architecture is represented by architectural elements such as services and connectors, deployment platforms (on-premise and cloud-based) and cloud services. The notation here is somewhat aligned with UML component diagrams. We have specific component types color-coded.

A service component can either be atomic or contain internal components allowing for hierarchical decomposition. For example, the migration pattern MP1 in Fig. 1 consists of a coarse-grained component that consumes services of an on-premise deployment platform. These can be coordination services that orchestrate different components in larger compartments or simply configurable IaaS resources providing required operating system or storage features. After migration, this component, instead of using on-premise platforms, uses services offered by a public cloud platform. Thus, the application component is re-deployed as-is on a cloud platform.

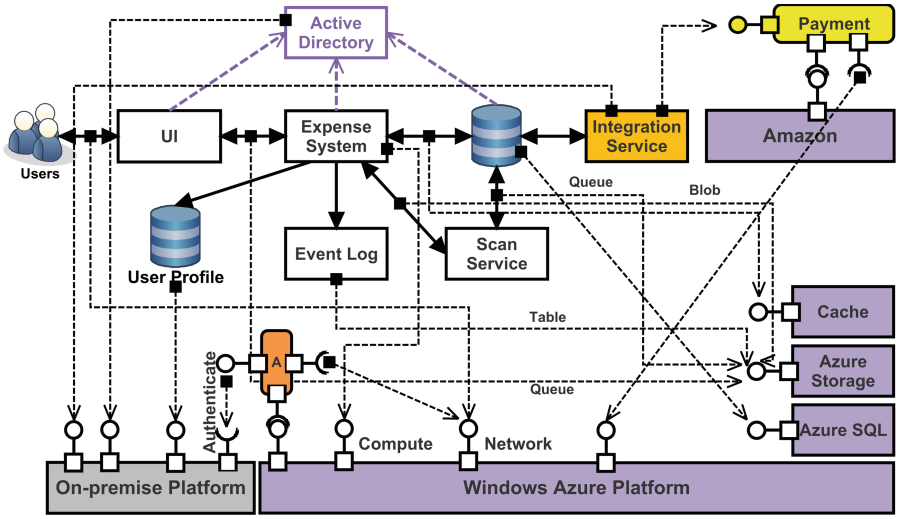

Fig. 4. Sample migrated architecture – after migration.

To identify a suitable pattern, the patterns are specified by pattern descriptions that focus on quality and resources/patterns to guide the selection process:

- Definition: A component re-hosted (or relocated) on a cloud platform is enhanced by using the environmental services of the other cloud platforms.
- Problem: Availability of an application needs to be enhanced without architecture change, and without capital expenditure for hardware.
- Solution: Leverage cloud platform environment services to improve availability, e.g., live migration from existing platform to target platform in case of outage.
- Benefits: As component re-hosting in multiple cloud platforms and improve availability and avoid vendor lock-in.
- Risks: Cloud providers do not provide the necessary services for applications to run in cloud platforms without re-architecting or rewriting code.

Our migration patterns are meant to be aligned sequences of architectural changes in the application, through which the current application is gradually refactored and modernized. For each migration pattern, an architectural migration schema has to be defined.

Our migration pattern catalogue comprises of 15 individual patterns. We have identified these 15 patterns through an empirical extraction process. To make the selection easier, these 15 patterns can be categorised into a number of Core Patterns and derived Pattern Variants. Full descriptions of core patterns and variants for the following core patterns can be found in detail in [26]. The categorisation is as follows:

- Re-deployment (core pattern MP1): variant pattern MP2 (for re-deployment in public cloud)
- Relocation (core pattern MP3): variant pattern MP4 (relocation for multi-clouds)

- Multi-Cloud Refactoring (core pattern MP5): variant patterns MP6 (hybrid refactoring), MP7 (hybrid refactoring with on-premise adaptation), MP8 (hybrid refactoring with cloud adaptation), MP9 (hybrid refactoring with hybrid adaptation)
- Multi-Cloud Rebinding (core pattern MP10): variant pattern MP11 (rebinding with cloud brokerage [23])
- Replacement (core pattern MP12): variant patterns MP13 (replacement with on-premise adaptation), MP14 (replacement with cloud adaptation)

Further variants can be added, but there is a sufficient completeness of the given set to model common PaaS migration scenarios, which we have demonstrated through past case study evaluations.

The core pattern and variants guides the migration pattern selection. Architecture aspects (from the application and platform profiles) and the technical quality constraints are the initial selection criteria. The pattern selection can be seen as a variability management problem that distinguishes internal (provider-based deployment) and external (application and application access) perspectives. To make this more clear, we can look at different applications. Some applications are integrated and support core business processes and services, but many of them support utility needs, are certainly non-core applications and are independent. The latter category may be obvious candidates for direct re-deployment. For the former integrated core ones, refactoring (re-architecting or redesigning) is more appropriate.

Migration paths emerge as sequential compositions of these patterns on a source architecture, see Fig. 1. These paths are defined based on discussions with the company about their existing architecture and a high-level specification of technical and business targets. Migration paths define decision points where typically several architectural options emerge, e.g., different data storage options [7–9]. For (a subset of) these options, an experimental evaluation can be considered.

2.2 Experimental Evaluation of Migration Options

The current architecture is ported into the cloud, but can there take advantage of virtualization to not only reduce operational expenditure, but also to create multiple instances of the application to improve scalability and failover without increasing capital expenditure. A risk is that underlying architecture concerns are not sufficiently addressed. A monolithic legacy application in the cloud is still monolithic including the previous limitations such as a lack of scalability. Scalability cannot easily be achieved if, e.g., the architecture does not allow the database to be updated by multiple instances.

We can used the pattern-based migration paths (including options) to investigate quality and cost concerns.

Based on the identified migration paths, a plan focusing on a subset of components is identified for experimental evaluation: Firstly, define source and possible target architectures. Secondly, select critical components, e.g., high volume data process to test scalability of storage (DB) or communications infrastructure to test integration and communications scalability. The benefit of the patterns is that they link architecture configuration to quality. We use this link to select components for the feasibility exploration based on the most relevant quality concern to be explored.

3 Architecture Assessment and Risk

This cloud migration process can be seen as an incarnation of a wider architecture modifiability method. We can benefit from a scenario-based approach to frame this, in particular one supporting the evaluation of modifiability. The migration patterns we introduced actually reflect different migration scenarios. An architecture analysis can inform the decision how migrateable (i.e., modifiable in a certain way) a system is and whether to migrate this. ATAM (Architecture Tradeoff Analysis Method) and ALMA (Architecture-Level Modifiability Analysis) are two widely used methods for architecture analysis [15]. Using these can help assessing the migration risk and identifying possible quality concerns that need to be further explored.

ATAM is not specifically positioned for migration evaluation, but it does more generally target the trade-offs between different quality concerns. However, a scenario-based approach as followed by ATAM is useful and can be complemented by architecture-level metrics that we use specifically for the migration evaluation.

ALMA is another scenario-based method, which is more specific to modifiability [15] and thus applies better in the migration context. ALMA is proposed for software architecture modifiability assessment by using a number of indicators: maintenance cost prediction, risk assessment. In case of assessing and comparing different system, the modifiability analysis performed with ALMA supports software architecture selection as well. ALMA is based on five steps, which we actually implement as part of the migration pattern method:

- Step 1: Goal definition using pattern properties.
- Step 2: Target architecture description using a pattern-based migration path.
- Step 3: Define (elicit) change scenarios. This means in our context to define migration plans, possibly involving different architectural alternatives (represented as alternative paths in a transition graph). The use of patterns allows us then to assemble alternatives from basic building blocks.
- Step 4: Evaluate scenarios, analyse expected and unexpected changes on a number of qualities. Examples of assessment criteria are cost and workload or performance to be experimentally evaluated. This is supported by properties attached to patterns.
- Step 5: Interpret results (pattern-based migration paths, annotated with quality properties).

The ALMA method allows us to consider maintenance cost and carry out in this way a form of risk assessment for a migration decision, which are the relevant concerns at the interaction of technical and business sustainability. Risks are clarified here through an experimental feasibility study, see Fig. 5.

We use the migration pattern to define a draft plan, to which we add here a risk assessment model of the target architecture options, driven by a selected architecture assessment method, such as ATAM or ALMA. The risk assessment introduces quality and cost metrics that can be integrated with the target architecture. The metrics include factors such as complexity and amount of changes on both of the architecture and source code, code efficiency and security. Those metrics change depending upon the existing software. However, the metrics also may change when correlated with business objectives and services costing model [25,29]. In order to answer those risk-based migration

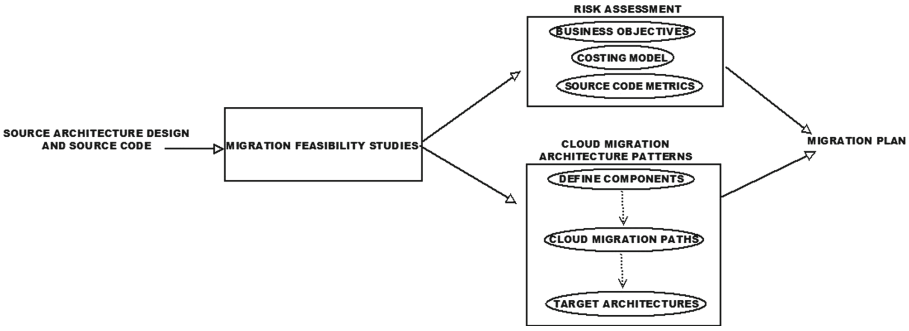

Fig. 5. Architecture and risk framework.

questions, a survey has been conducted by us about the business needs and costs. The risk assessment aims to address the issues that could occur in the migration process, as well as designing more reliable plan in terms of resources and time.

- Defining the metrics: defining what we aim by each factor since those factors could be utilized in different way:
  - Efficiency metrics: indicate how many methods, modules will be called and executed that are included in scaling cycle iterations, e.g., how far the application scales, how many times it was used.
  - Complexity metrics: indicate the amount of effort needed to develop and maintain the code. The more difficult code, the more effort and time needed to develop and maintain this code. The code metrics calculates: Maintainability Index, Cyclomatic Complexity, Depth of Inheritance, Class Coupling and Lines of Code.
  - Security metrics: security requirements that indicate vulnerability, violations, and threats with perspectives of Authentication, Authorisation and Storage Security. However, security practices and related metrics is beyond the scope here.
- During experimentation: we apply the new metrics approach on our use cases, such as performance and scalability concerns.
- Understanding the new architecture components: a way to improve the scalability and efficiency of the existing software is to refactor the software architecture to have more decoupled and separated service components.

From the different metrics listed here, we will largely focus on efficiency aspects in the use cases that will be introduced later.

4 Experimentally-Oriented Migration Feasibility Studies

Experimentation is the most important activity in the proposed feasibility study approach. We will look into objectives and the role of experiments to determine feasibility. Some use cases will illustrate the concerns.

4.1 Quality Requirements and Possible Cost Implications

The feasibility studies are driven by the following goals:

- Quality and Cost: This concerns the quantification of quality and cost aspects. Often, scalability is an important concern, driven by the business objective to expand. In this case, an experimental feasibility study can validate a proposed architecture scalability. Another motivation is a cost-vs-performance experiment, i.e., to consider different options and compare them technically (e.g., scalability of different target architecture options), but rank them considering the costs that they would incur [14, 22].
- Usage and Cost: By looking at the usage, we change the focus to the potential user. Usage exploration through experiments is a suitable tool to explore usage patterns and predict potential income based on this. This can then directly be related to the resources (and their costs) to facilitate user requests.
- Process and Platform Understanding: Experimentation allows to achieve a better understanding of technical constraints and operational activities in the cloud. What experimentation can shows is the difference between PaaS/IaaS/SaaS solutions (as consumer and provider) and integration and interoperation problems. It also clarifies how to structure and cost a staged migration (plan derivation).

The problem that emerges in the migration decision and planning process that links quality, usage and costs to the architectural configuration, can be phrased in the following question: *How many processes can be hosted on a fixed cloud compute resource with a pre-defined latency performance target for a forecasted number of users of a particular application with a forecasted mix of application operation usage.*

4.2 Experimental Studies as a Solution for Quality and Cost Estimation

Experimental studies can play an important role in the determination of migration feasibility. Experimentation often results in a prototype evaluation of a partly cloud-native architecture. Rather than just cloudifying a system in a virtual machine, we often selected a component such as data storage and have experimented with different cloud-native storage options, including for instance a mix of traditional RDBM and other table/blob storage formats. Partial experimentation with cloud-native prototypes allows to consider a fully cloud-native architecture to be discussed with realistic technical (e.g., scalability) and cost assumptions (storage, access) [10]. Only realistic costs for cloud operation allows a charging model to be developed and validated that fits their own product.

4.3 Use Cases

We discuss the migration concerns and how they are documented. When looking at concrete use cases, we distinguish what is expected and understood on the one hand and misconceptions and lack of knowledge on the other.

- Clarity of expectations and objectives: Business reasons to go to cloud are often clear, e.g., a planned internationalisation or expecting an increase in company value (in the cloud). Technical reasons for a cloud migration are at a high level clear, for instance scalability, but often lack deeper understanding of cost and quality.
- Understanding of cloud concerns (having an impact on architecture and process selection) is often limited. Technically, the difference between provision models/layers is unclear. This includes the management effort at I/P/SaaS level in comparison. Another problem is a possible vendor lock-in. In business terms, e.g., possibly required revenue model changes remain very unclear. Legal/Governance concerns such as data location are known, but without reliable knowledge.

We have conducted a number of case studies in the following application domains. We highlight the specific needs for a feasibility study that has emerged in each of the cases.

The central case study, which we will also use later for further illustration is *document management*: the application is document image processing to allow more efficient processing in the cloud, but also to enable the company to extend into new markets.

The other relevant use cases are:

- *Banking solution:* an integrated solution (account management plus ATM operations) – provided in Africa and Asia, raising uncertainty concerns from security to legal,
- *Insurance:* a solution for multi-product management in multiple countries – uncertainty arise from the need for variability management of a single product across different regions/jurisdiction,
- *Food sector ERP:* an ERP solution for food production and sales – where a stable in-house solution is prepared for launch as a product into different markets. Food safety regulations impact on the architecture a cloud-based solution,
- *Business Registry:* enterprise repositories – scalable internationalisation is the driver, allowing clients to access their services through the cloud.

5 Structured Experimentation in Migration Feasibility Studies

We now explore the proposed feasibility studies based on experimentation in more detail. Experimentation aims to allow to establish a link between technical feasibility and quality (e.g., performance) and costs [7]. The pattern-approach helps to guide the select the components for exploration based on the most relevant quality concern. This can range from performance to security to integration and interoperability, as indicated for the use cases.

Furthermore, in many development approaches, quality testing is an integral component, in particular if the software is directly used by end-users. Load testing is difficult if prior testing has been done in-house and no expertise in testing in the cloud exists. The feasibility study thus also addresses load testing.

5.1 A Process for Migration Feasibility Studies

We frame our experimental approach in a process for migration feasibility studies:

1. Definition of the source architectures and a number of possible target architectures,
2. Selection of critical components, e.g., a high volume data process to test scalability of storage (DB) or a communications infrastructure to test integration or communications scalability.

This is a critical challenges for the migration expert and the company architects. They have to understand the existing architecture and to select the most suited component for the experimental studies (ranges from individual component to full virtualisation as VM). The feasibility process is based on architecture determination, prototype component selection, and construction of realistic use case application for further analysis: (i) setup of services and data (real or dummy) and monitoring, (ii) experiment specification, (iii) experiment execution, (iv) data collection and analysis.

5.2 Benefits of Experimentatal Studies

Some of the experiments that we carried out targeted for instance data storage for image processing in the 'Document Management' use case. What experimentation shows are a number of concerns that would normally not always be identified and clarified in discussions and non-experimental analyses:

– Difference between PaaS/IaaS/SaaS solutions (as consumer and provider). Based on the migration architecture it allows to practically demonstrate configuration and operation of different scenarios.
– Scalability of the solutions, as exemplified in Figs. 6 and 7 for response time behaviour based on different retrieval and update loads.

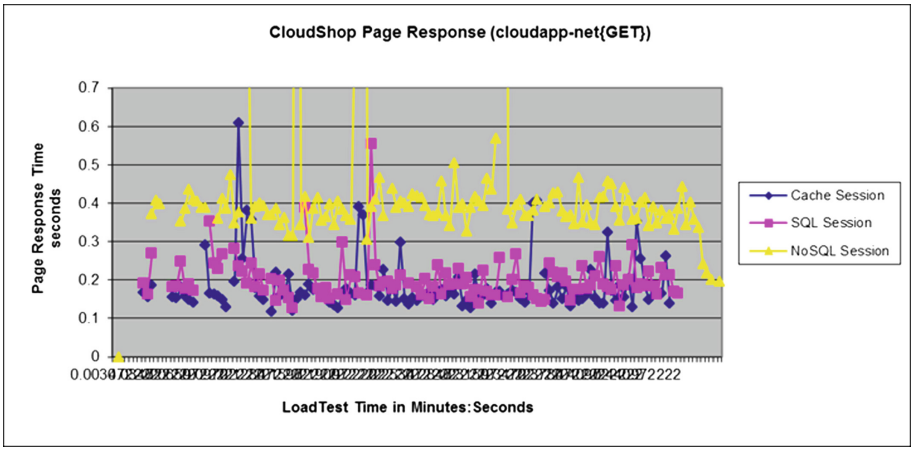

Fig. 6. Comparison of storage options (technical quality). Three DB configurations (Cache, SQL, NoSQL) are compared in terms of the page response time (in sec) [1] for given load test times.

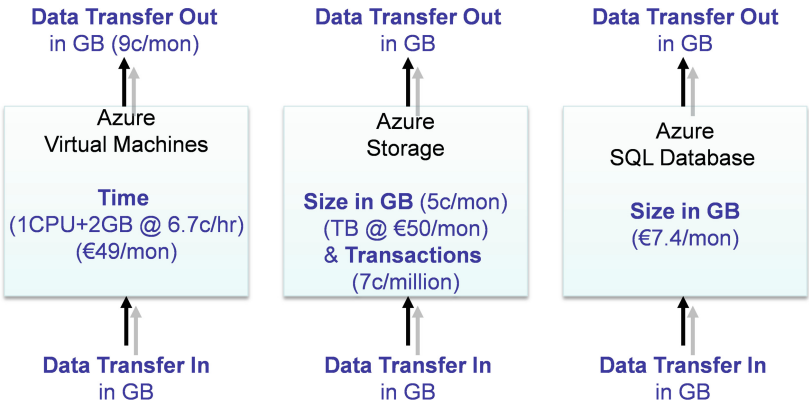

Fig. 7. Comparison of storage options (data consumption versus cost) [1].

- Identification of integration and interoperation problems that emerge in the prototype implementation when on-premise components are migrated.
- How to structure and cost a staged migration (plan derivation) can be estimated on the prototype component migration that typical involves the migration of existing components. In order to experiment, data and traffic needs to be available, again either imported or generated. In all cases, a realistic prediction of software and data migration tasks and related costs is possible.

Figure 6 shows the performance results for different storage options for a Web application. Figure 7 maps the architectural options onto costs.

5.3 Costs of the Experimental Effort

Cost is a key factor in the decision whether to conduct a feasibility study, i.e., does it pay off to carry out a feasibility study. The costs include the cloud platform to experiment with and the migration expert.

The total experimental costs are 5–10% of the predicted migration costs based on our case study experience – including architectural analysis, draft migration plan and security and data protection analysis[1].

6 A Selected Migration Use Case

In the migration process, the problem is costing an application on a public cloud within the given performance and scalability quality requirements. The total costs include the implementation costs of refactoring the system as a cloud application, as well as the operational cloud charges associated with running the new system on a cloud provider.

[1] Please note, these costs were part-funded for the given use cases documented here by governmental business innovation schemes.

This use case – the document management use case – focuses on the migration of a legacy client-server on-premise single-tenant enterprise application to the cloud by re-designing, re-engineering and recoding the system as a cloud application. The company in question is an independent software vendor (ISV) that has over 1,000 existing client installs and this case study is to estimate the costs in migrating a part of these to a cloud platform. The application is Document Management, which enables scanning documents and save them as electronic images. Documents can be classified and metadata templates are used to store searchable tagged data for the documents for retrieval and reporting. Additional services such as image processing, which allows for example Optical Character Recognition (OCR) and barcode reading can be added. The application is a multi-process system that consists of a web server (a compute resource) and a separate image processing component (another compute resource). However, the functional dependency between these does not need to be considered in any cost analysis as image processing VMs work independently from the web server, which operates regardless of the state of the image processor. Therefore, we can calculate VM configuration requirements based on a linear multiplication of the CPU load per tenant.

For the experimental part of this use case, we have considered the following architecture components:

- A cloud data store – made up of Azure-based NoSQL Table structure (Azure Table service) and an object store (Azure Blob Storage service). The former service stores metadata associated with the documents. The latter service stores the corresponding document image files.
- A cloud compute architecture made up of a separate compute resource for the web server of the web application (Web Role Virtual Machine), and a separate compute component for carrying out the image processing functions, such as barcode reading (Worker Role Virtual Machine);

The following data sources of data were used for an estimated technical configuration and cost calculation of a suitable full system.

- Prediction of the usage of storage resources when applications and their tenant are migrated: if available, we can use actual historical data from an existing average-sized tenant with a typical application usage pattern. The historical storage information, available from experiments or related sources, namely, the images, metadata and template files saved by a user of the application a given period in an on-premise context, provided a reference data store usage profile for a typical average user of data stored on the cloud.
- Prediction of the cloud compute resources required: we can monitor the usage and performance during an experimental period of the operational use of the application, again by a typical user as above. Monitoring includes here the application functions called and the number of document images/metadata stored and modified.
- In order to support the configuration to select the optimal VM type, we can carry out a benchmark study of the performance of the different VMs as part of the experimentation.

For the given use case, an experimental benchmarking and performance analysis has identified that the VM CPU load is the driving factor that determines the compute

resource required. Therefore, this parameter is used as the main driver in the calculation of cloud costs in relation to performance requirements.

As an observation on the experiments carried out, there is an important difference between storage and compute resources in the cost calculation.

- When the cloud provider provides automatic scaling of the available storage capacity so that it appears to be effectively infinite from the application perspective. The application or cloud orchestration do not need to consider scaling storage capacity.
- The same is not the case for the compute resources. The auto-scaling of Virtual Machines (VM) must be configured and monitored by the administrator and set according to changing tenant numbers and sizes and application usage patterns and operational profiles. It must be carried out by taking Quality-Of-Service (QoS) parameters and Service Level Agreement (SLA) targets into account.

As a consequence, we monitored the usage and performance of the application over a test period during which a user carried out some typical operations of the application. We measured VM CPU and Memory utilisation as well as web application page response times and storage access end-to-end latency times. In the use case, the measured response times have demonstrated that the performance for file searching and scanning are within acceptable limits. Note that storage latency times are outside the control of the application and are a quality issue with the cloud provider, such as Microsoft Azure in the given case. The Memory and CPU usage indicated that the dedicated image processing VM is constant. This was expected since the processing was managed by an Azure Queue service which spreads the workload suitably. The memory utilisation of the server is equally constant, which can also be expected for a web server running web applications such as this.

While these observation are helpful in the configuration and cost calculation, we need to note some caveats. The calculations carried out in the experimental phases were to some extent rudimentary, because they were carried out before more comprehensive performance tuning could have been performed on the full application and the platform it was put on. The following aspects of the application and cloud platform could be further assessed during full operation of the application, e.g., in order to improve and optimise the deployment of the application.

- The existing deployment does not include any data caching, which would likely significantly reduce the CPU overhead as well as data storage access costs.
- The platform can be re-engineered to make use of newer platform services which allow for code to be run on demand rather than on compute role services – cf., recent lambda functions available by some major cloud service providers that should significantly reduce compute costs.
- The CPU load can be further optimised by looking at the queries to the table service through the use of indexing tables or schema denormalisation. This could be carried out once the more heavily used operational queries have been identified.

It should also be noted that a simple linear multiplier has been used here to estimate the tenant load for the forecasted tenant numbers. This, as explained does not take into account of the smoothing effect of multiple tenants sharing the same application

compute resources. It therefore represents a conservative estimate of compute VMs required.

Detailed results on the effectiveness of the experimental feasibility studies will be reported in the next section.

7 Observations on Use Cases and Evaluation

The benefits of feasibility studies in the proposed format are experimentation results and documentation:

- a proper documentation of scenarios and plans needed at a high level, which allow detailed migration plans for a full migration to be developed,
- experiments help to clarify options, address misconceptions, and identify open problems.

They help to scope a full migration project. We have evaluated our feasibility approach to cloud migration analysis and planning based on the following criteria:

- Clarification of architectural options ans concerns.
- Reliable quality prediction & cost prediction.
- Cost effectiveness as an instrument.

7.1 Clarification of Architectural Concerns

A clarification of architectural options for organisations planning to migrate through architectural prototyping using a pattern approach allows to use patterns that link quality to structure [16, 19]. A survey has been carried out with architects involved in the migration studies – with at least 3 architects for each participating use case. This includes architects both within the company in question and also consulting software architects.

The key results of the survey we carried out are:

- All participants surveyed (100%) agree or strongly agree that the method is suitable as an analysis tool to identify options and concerns.
- Almost all (88% of the participants) agree that the migration method is suitable to analyse and discuss functional and non-functional architecture requirements for migration.

However, there has been one limitation that has been flagged by the survey participants. While 55% strongly agree that the method is suitable for SMEs and that is also suitable for multi-cloud migration (80% positive), there are also 43% that have concerns with its applicability for large-scale migration. This remains a concern for future work.

7.2 Prediction of Software Quality and Cost

We have already talked about the link between quality and cost. We can distinguish quality and cost observations in an attempt to obtain reliable estimations:

- Reliable Quality Prediction: experimental quality assessment results in trustworthy, reliable input to predict full system behaviour. This has been discussed and analysed with the customer companies in question. In the case of successful migrations at a later stage, we have not found any major deviations from the predictions when using the configurations experimented with.
- Reliable Cost Prediction: experimental cost assessment provides a low-to-high demand cost range, allowing to predict to system costs on a reliable basis [11]. These have been discussed with the customer company and, where possible, evaluated through a later full implementation.

Only an empirical evaluation of the two quality items is currently possible due to the lack of suitable quality and cost mapping models. Cloud computing is here, however, highly suitable as monitoring of technical details can be set to be detailed for the experiments and billing for the resources used is equally detailed with typically metered usage for different resources used. For instance, for the 'Document Management' use case, the factors considered for storage only were: *region, replication options, data size stored, transaction number, data transfer*.

7.3 Cost Effectiveness of Migration Feasibility Studies

Cost effectiveness as an instrument is a key aim of our approach. We looked at the actual costs of feasibility studies relative to full migration costs. We also considered how successful feasibility studies were in terms of determining a decision for the actual migration.

- Cost per project: For the 5 migration cases that we have carried out, the average cost was around 5% of the total budget for a full migration. The full migration budget was determined after the feasibility study concluded. The 5% cost was considered adequate. Please note that the overall budget for the migrations ranged between 100,000 and 250,000 Euros, including substantial re-architecting in some cases.
- Decisions taken: Concrete results from the use cases are as follows: 1 full migration (document management) has started and is currently being finalised, 1 full migration (insurance) has started, 1 decision against due to quality grounds (banking), 2 decision have at this stage not yet been taken (food, registry). In the all finalised cases, the companies in questions have based their decision on the outcome of the feasibility studies.

For the discussion with the companies in question, readiness to make a decision to embark on a full migration is considered a success. The experimental feasibility study is also considered successful if a decision against migration is taken on the grounds of an unfavourable quality or cost result.

7.4 Discussion of the Effectiveness of Migration Feasibility Studies

The limitations of the approach are related to the delay it might cause. In many circumstances this might not be an issue, but in some situations an approval to embark on a full migration is necessary, possibly depending on a successful outcome of the study. The format of this approval is typically one of the following:

– External private investment in a cloud
– Public support, e.g. through innovation schemes
– Internal approval for further funding

While there might be a delay particularly if an approval process is included, any full migration decision can be based on reliable input data. Please note that in the use cases where a decision has not been taken yet, this was not due to the study results.

In the context of the whole migration costs and generally some risk involved, the low feasibility study cost are a valuable investment.

Another limitation at the moment is that we have only carried out migration studies for small to medium sized applications with SMEs as customers. Data for large-scale experimentation with a possibly more complex setup does not exist yet.

8 Related Work

We distinguish related work in terms of three concerns – architecture migration, general cloud costing and quality-linked costing and pricing.

Architecture Migration. A number of cloud migration approaches exist. Authors like Jamshidi et al. [2] survey the literature on cloud migration. Fahmideh et al. [18] provide a comparison framework for cloud migration methods. We target specifically an experimental framework that goes beyond the surveyed process models for migration. However, we assume here a pattern-based migration method, into which a number of existing methods fall. To the best of our knowledge, the role of feasibility studies has not been explored yet in the context of cloud migration.

Costing in Migration. Costing models for cloud are important for organisations to understand their own costs and expenses. In [5], an overview of pricing models for the cloud for operational costs is provided. On the expenses side, e.g., at IaaS level, resources are priced often like commodities [11]. At the income side for an ISV operating through the cloud, the product is typically provided as a SaaS with possibly a mix of model from pay-per-use to pay-per-user and flat-fee models. Our solution is meant to bridge between the two perspectives.

Another direction would be the consideration of the total cost of ownership (TCO) and the Return-On-Investment (ROI) [32]. TCO in a cloud context would include the migration costs as well as the operational costs of running an application in the cloud. We were primarily interested in the operational costs for given quality requirements. Thus, actual migration costs for re-engineering and adaptation have been ignored in this investigation. Our work could be extended by determining the TCO for an SLA-compliant configuration.

Architecture and Pricing. Quality in the cloud is manageable for quality factors as performance by configuring and adapting the virtual resources used appropriately. We propose a manual experimentation approach for cloud prototype implementation. In [4] a system to support automated resource selection is suggested. Although this is not generally applicable, the ideas could help to automate our approach further. The automation of cloud experimentation is also addressed in [20] through a tool suite for the OpenStack platform.

9 Conclusions

The decision whether to run software in the cloud is both a cost and a quality question. For instance, fior a cloud-based software provider, quality and cost need to be reconciled in a cloud-based system architecture. This architecture needs to map software hosted using IaaS or PaaS services onto a SaaS delivery model. In this context, the reliability of relevant data and an understanding of the processes of migration and operating in the cloud and their impact are important elements for taking a decisions [17].

Our proposal here is to support this process through feasibility studies. These can help the companies in question to determine or at least confidently estimate the costs of the cloud migration, but also the operation of a software product in the cloud. Feasibility studies provide decision support. The key questions are whether and how a software product can be deployed and delivered cost-effectively in the cloud while at the same time maintaining desired quality. The benefit is increased reliability of data/assumptions, rather than relying on experience or guesses. We put experiments at the core of these feasibility studies. Testing and experimentation is often not don until a much later stage around systems deployments. Performance is the focus of these experiments. Load tests are normally not done a performance testing stage. In our proposal, load testing is an experimental technique that allows to reduce technical risks before the actual migration starts.

We have demonstrated the costs of conducting a feasibility study are moderate given the risks of failure during a full migration or during the operation of the software in question.

The experimental part relies currently on the manual setup by an experienced consultant. This could be improved through an automation of the experiments. What could help is automated test case generation for performance and scalability. The test configuration could also consider alternative configurations, such as an automated storage service selection and configuration. Also relevant is the consideration of large-scale application migration. For instance, it is currently unclear if migration costs scale up linearly based on the application size or if normal development cost heuristics can be applied. A further possible direction is to focus on prevalent architectural trends such as cloud-native microservice [24,30,31] or container-based architectures [27]. These architectures can be reflected in the patterns themselves, for instance more clearly identifying cloud-native architectures in terms of a microservices style [21]. This would allow to fine-tune settings in terms of performance [28].

Acknowledgements. This work was partly supported by IC4 (the Irish Centre for Cloud Computing and Commerce), funded by EI and IDA.

References

1. Fowley, F., Elango, D.M., Magar, H., Pahl, C.: The role of experimental exploration in cloud migration for SMEs. In: International Conference on Cloud Computing and Services Science, CLOSER 2017 (2017)
2. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)

3. Jamshidi, P., Pahl, C., Mendonca, N.C.: Pattern-based multi-cloud architecture migration. Softw. Pract. Experience **47**(9), 1159–1184 (2016)
4. Son, J.: Automated Decision System for Efficient Resource Selection and Allocation in Inter-Clouds. The University of Melbourne (2013)
5. Arshad, S., Ullah, S., Khan, S.A., Awan, M.D. and Khayal, M.: A survey of Cloud computing variable pricing models. In: Evaluation of Novel Approaches to Software Engineering (2015)
6. Jamshidi, P., Pahl, C., Chinenyeze, S., Liu, X.: Cloud migration patterns: a multi-cloud service architecture perspective. In: Toumani, F., Pernici, B., Grigori, D., Benslimane, D., Mendling, J., Ben Hadj-Alouane, N., Blake, B., Perrin, O., Saleh, I., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8954, pp. 6–19. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22885-3_2
7. Xiong, H., Fowley, F., Pahl, C., Moran, N.: Scalable architectures for platform-as-a-service clouds: performance and cost analysis. In: Avgeriou, P., Zdun, U. (eds.) ECSA 2014. LNCS, vol. 8627, pp. 226–233. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09970-5_21
8. Pahl, C., Xiong, H.: Migration to PaaS clouds - migration process and architectural concerns. In: MESOCA Symposium (2013)
9. Pahl, C., Xiong, H., Walshe, R.: A comparison of on-premise to cloud migration approaches. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 212–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40651-5_18
10. Al-Roomi, M., Al-Ebrahim, A., Buqrais, S., Ahmad, I.: Cloud computing pricing models: a survey. Int. J. Grid Distr. Comp. **6**(5), 93–106 (2013). https://doi.org/10.14257/ijgdc.2013.6.5.09
11. Wang, W., Zhang, P., Lan, L., Aggarwal, V.: Datacenter net profit optimization with deadline dependent pricing. In: Conference on Information Sciences and Systems (2012)
12. Giardino, C., Bajwa, S.S., Wang, X., Abrahamsson, P.: Key challenges in early-stage software startups. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) XP 2015. LNBIP, vol. 212, pp. 52–63. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_5
13. Li, H., Zhong, L., Liu, J., Li, B., Xu, K.: Cost-effective partial migration of VoD services to content clouds. In: Cloud Computing (2011)
14. Pahl, C., Jamshidi, P., Weyns, D.: Cloud architecture continuity: change models and change rules for sustainable cloud software architectures. In: Proceedings IEEE 4th International Conference on Cloud Computing, pp. 203–210 (2011). https://doi.org/10.1109/CLOUD.2011.41
15. Koziolek, H.: Sustainability evaluation of software architectures: a systematic review. In: Joint ACM Symposium on Quality of Software Architectures QoSA and Architecting Critical Systems ISARCS, pp. 3–12 (2011)
16. Pahl, C.: Layered ontological modelling for web service-oriented model-driven architecture. In: Hartman, A., Kreische, D. (eds.) ECMDA-FA 2005. LNCS, vol. 3748, pp. 88–102. Springer, Heidelberg (2005). https://doi.org/10.1007/11581741_8
17. Chappell, D.: Cloud Computing White Papers (2016). http://www.davidchappell.com/writing/white_papers.php
18. Gholami, M.F., Daneshgar, F., Rabhi, F.: Cloud migration methodologies: preliminary findings. In: CloudWays Workshop (2016)
19. Pahl, C., Lee, B.: Containers and clusters for edge cloud architectures - a technology review. In: International Conference on Future Internet of Things and Cloud (2015)
20. Affetti, L., Bresciani, G., Guinea, S.: aDock: a cloud infrastructure experimentation environment based on open stack and docker. In: International Conference Cloud Computing (2015)
21. Pahl, C. Jamshidi, P., Zimmermann, O.: Architectural principles for cloud software. ACM Trans. Internet Technol. **18** (2017). Article no. 17

22. Fowley, F., Pahl, C., Zhang, L.: A comparison framework and review of service brokerage solutions for cloud architectures. In: Lomuscio, A.R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I. (eds.) ICSOC 2013. LNCS, vol. 8377, pp. 137–149. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06859-6_13
23. Fowley, F., Pahl, C., Jamshidi, P., Fang, D., Liu, X.: A Classification and Comparison Framework for Cloud Service Brokerage Architectures. IEEE Trans. Cloud Comput. **6**(2), 358–371 (2018). https://doi.org/10.1109/TCC.2016.2537333
24. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations and issues for migrating to microservices architectures: an empirical investigation. IEEE Cloud Comput. **4**(5), 22–32 (2017)
25. Pahl, C., Giesecke, S., Hasselbring, W.: Ontology-based modelling of architectural styles. Inf. Softw. Technol. (IST) **51**(12), 1739–1749 (2009)
26. Jamshidi, P., Pahl, C., Mendonca, N.C.: Pattern-based multi-cloud architecture migration. Softw. Pract. Experience **47**(9), 1159–1184 (2017)
27. Pahl, C., Brogi, A., Soldani, J., Jamshidi, P.: Cloud container technologies: a state-of-the-art review. IEEE Trans. Cloud Comput. (2017). https://ieeexplore.ieee.org/document/7922500/
28. Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L.E., Pahl, C., Schulte, S., Wettinger, J.: Performance engineering for microservices: research challenges and directions. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (2017)
29. Javed, M., Abgaz, Y.M., Pahl, C.: Ontology change management and identification of change patterns. J. Data Semant. **2**(2–3), 119–143 (2013)
30. Pahl, C., Jamshidi, P.: Microservices: a systematic mapping study. In: Proceedings CLOSER Conference, pp. 137–146 (2016)
31. Aderaldo, C.M., Mendonca, N.C., Pahl, C., Jamshidi, P.: Benchmark requirements for microservices architecture research. In: Proceedings of the 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering. IEEE (2017)
32. ISACA. Calculating Cloud ROI: From the Customer Perspective (2012). https://www.isaca.org/knowledge-center/research/researchdeliverables/pages/calculating-cloud-roi-from-the-customer-perspective.aspx